DESIGN OF A SYSTEM FOR MULTIPLE ROUTE SELECTION IN THE PRESENCE OF FLOODING

A Thesis Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Priyal Prasanna Kulkarni

August 2018

DESIGN OF A SYSTEM FOR MULTIPLE ROUTE SELECTION IN THE PRESENCE OF FLOODING

Priyal Prasanna Kulkarni

APPROVED:

Dr. Christoph F. Eick, Chairman
Dept. of Computer Science

Dr. Ioannis Pavlidis
Dept. of Computer Science

Dr. Suresh Khator
Dept. of Industrial Engineering

Dean, College of Natural Sciences and Mathematics

Acknowledgements

I would like to express my gratitude to my thesis advisor Dr Christoph F. Eick for the useful comments, remarks and engagement through the learning process of this Master's thesis. I would also like to thank the experts who were a part of the committee for this thesis: Dr Ioannis Pavlidis and Dr Suresh Khator for their passionate participation and inputs.I would like to thank the incredible team of Data Analysis and Intelligent Systems Laboratory for their continual insights and motivation during the course of this thesis.

Finally, I must express my profound gratitude to my parents, dear brother and loving friends for providing me with unfailing support and continuous encouragement throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you!

DESIGN OF A SYSTEM FOR MULTIPLE ROUTE SELECTION IN THE PRESENCE OF FLOODING

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Priyal Prasanna Kulkarni

August 2018

Abstract

Safe navigation during flooding is integral in minimizing loss of life. Navigation has throughout literature been treated as a search problem with the aim of optimizing certain impedance. The earliest study of path finding started in the late 1800s forming the basis of depth-first search techniques. This was followed by the introduction of a lot of popular algorithms including Dijkstra's, A*and Bellman-Ford. Recently, the study of path planning for road networks based on heuristics for dynamic or partially known environments has gained a lot of attention. In this thesis, we present a unique approach to finding multiple competitive paths between two locations on a street network that also considers road flooding. The key idea is to find a cost optimal solution for two locations using Dijkstra's algorithm. We then penalize the found solution by increasing the traversal cost of one segment or the whole path, forcing the search algorithm to find alternative solutions. This framework is developed for the street network the City of Houston leveraging the capabilities of ArcGIS Desktop and Python scripting. The proposed algorithm is evaluated for quality and safety of resultant routes. This is done by comparing route lengths, elevations, widths, percentage of duplicate road segments, maximum speed limits of the obtained paths. We also conducted an experimental evaluation that shows an elevated sensitivity towards these factors as compared to the standard shortest path algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Shortest Path Problem is a classic problem studied in graph theory. Graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices, nodes, or points which are connected by edges, arcs, or lines[23]. Some of the most common shortest path algorithms include Dijkstra's, A*, Bellman- Ford, Floyd-Warshall, Genetic Algorithms and Johnsons algorithm.

Transportation across a road network is typically studied as a graph traversal problem. The road segments are represented as graph edges and the road intersections are represented as graph nodes. In general, a path between two network graph points exists if there is a set of nodes and edges that connect the two points. The problem of finding all paths between two points on such a graph is considered a NP-hard problem due to the cyclic nature of network graphs. Thus, most popular algorithms try to find the shortest path between these two points. A road network

can be considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment, or the cost of traversing the segment. Using directed edges, it is also possible to model one-way streets. Such graphs are special in the sense that some edges are more important than others for long distance travel (e.g., highways). This property has been formalized using the notion of highway dimension[24]. Some of the most common algorithms used for graph traversals for road networks include ALT (A* search, landmarks, and triangle inequality), Arc flags, Contraction hierarchies, Transit node routing, Reach-based pruning, and Labeling algorithm[1].

A popular methodology of modelling road networks is in the form of a Geographical Information System (GIS). A geographic information system (GIS) is a framework for gathering, managing, and analyzing data. Rooted in the science of geography, GIS integrates many types of data. It analyzes spatial location and organizes layers of information into visualizations using maps and 3D scenes. With this unique capability, GIS reveals deeper insights into data, such as patterns, relationships, and situationshelping users make smarter decisions [10]. Several frameworks offering GIS capabilities are available today. The most popular ones include AutoCAD Map 3D, ArcGIS, Google Earth Pro, Mapbox, Google Maps API, and ArcGIS Online. A lot of transportation related applications are developed using ArcGIS. ArcGIS has industry leading modelling of complex road networks. ArcGIS Network Analyst provides network-based spatial analysis tools for solving complex routing

problems. It uses a configurable transportation network data model, allowing organizations to accurately represent their unique network requirements. Routes for an entire fleet, calculate drive-times, locate facilities, and solve other network related problems are popularly solved using the Network Analyst [7]. However, Network Analyst fails to provide alternative paths between two points across a network. This is especially important during times when one or more roads are expected to have closures due to either constructions or certain disasters such as Flooding. Moreover, it is important to balance the allocation of vehicles to different routes to minimize congestion. A major portion of the study of routing across transportation networks focuses on optimizing certain attribute like minimizing travel time, minimizing total distance, minimizing the number of turns, maximizing road widths, etc. However, this approach to routing provides no effective insights during disasters and disaster based evacuations.

In our approach we develop an intelligent system that can provide multiple route options given a start and end location. The key idea is to find a cost optimal solution for two locations using Dijkstra's algorithm and rerun the algorithm by increasing the traversal cost of one segment or the whole path, forcing the search algorithm to find alternative solutions. Once multiple routes are obtained, our approach additionally evaluates the quality and safety of resultant routes. This is done by comparing route lengths, elevations, widths, percentage of shared road segments, maximum speed limits of the obtained paths. Multiple routes during evacuations enable in directing vehicles to different routes. The proposed approach also uses elevation maps to annotate the roads. This allows routing to consider elevation constraints that helps

provide safer routes during flooding.

The remainder of the thesis is organized as follows: Chapters 2 and 3 introduce the background and the related work, respectively. Chapter 4 gives a detailed explanation of our approach to obtain multiple routes. In Chapter 5, we provide a detail description of the components of ArcGIS used in the system implementation. These are parts of the preprocessing steps, the actual algorithm and the post processing of results. In Chapter 6, we test our algorithms for the road network of the City of Houston and provide the results of those experiments. Chapter 7 provides a conclusion of this thesis.

Chapter 2

Background

The framework developed in this study is based on the fundamentals of GIS or Geographical Information System. A geographic information system (GIS) is a framework for gathering, managing, and analyzing data. Rooted in the science of geography, GIS integrates many types of data. It analyzes spatial location and organizes layers of information into visualizations using maps and 3D scenes [10]. In this chapter we first study GIS data and its various building blocks. The software used in this study is ArcGIS for Desktop version 10.5.1 by Environmental Systems Research Institute (popularly known as ESRI). The chapter further discusses the various components of ArcGIS relevant to the study are discussed in detail. We also mention the data sets that form the knowledge base of the framework.

Figure 2.1: GIS Components

2.1 GIS and its Components

Using a GIS involves complete understanding about the methodology, patterns and processes needed to solve a problem. Figure 2.1 lists the five major components of every GIS system[18]. This includes the Hardware and software for processing, People, who decide the right hardware and software to be used and Analysis which helps in extraction of meaningful information from the available data. The most important component of a GIS system however, is the Data.

GIS data can be separated into two categories: spatially referenced data which

are represented by vector and raster forms (including imagery) and attribute tables which are represented in tabular format. Within the spatial referenced data group, the GIS data can be further classified into two different types: vector and raster. Most GIS software applications mainly focus on the usage and manipulation of vector geodatabases with added components to work with raster-based geodatabases[2].

Vector data is split into three types: polygon, line (or arc), and point data. Polygons are used to represent areas such as the boundary of a city (on a large scale map), lake, or forest. Polygon features are two dimensional and therefore can be used to measure the area and perimeter of a geographic feature. Polygon features are most commonly distinguished using either a thematic mapping symbology (color schemes), patterns, or in the case of numeric gradation, a color gradation scheme could be used[2].

Line (or arc) data is used to represent linear features. Common examples would be rivers, trails, and streets. Line features only have one dimension and therefore can only be used to measure length. Line features have a starting and ending point. Common examples would be road centerlines and hydrology. Symbology most commonly used to distinguish arc features from one another are line types (solid lines versus dashed lines) and combinations using colors and line thicknesses. In the example below roads are distinguished from the stream network by designating the roads as a solid black line and the hydrology a dashed blue line[2].

Point data is most commonly used to represent nonadjacent features and to represent discrete data points. Points have zero dimensions, therefore you can measure neither length nor area with this dataset. Examples would be schools, points of

interest, and in the example below, bridge and culvert locations. Point features are also used to represent abstract points. For instance, point locations could represent city locations or place names[2]. Raster data (also known as grid data) represents the fourth type of feature: surfaces. Raster data is cell-based and this data category also includes aerial and satellite imagery. There are two types of raster data: continuous and discrete. An example of discrete raster data is population density. Continuous data examples are temperature and elevation measurements[2]. There are also three types of raster datasets: thematic data, spectral data, and pictures (imagery).

2.2 ArcGIS for Desktop

ArcGIS Desktop includes a suite of integrated applications, including ArcMap, ArcCatalog, and ArcToolbox. By using these applications and interfaces in unison, any GIS task including mapping, geographic analysis, data editing and compilation, data management, visualization, and geoprocessing can be performed with ease. ArcMap is the central application in ArcGIS Desktop. It is the GIS application used for all map-based tasks, including cartography, map analysis, and editing. ArcMap offers different ways to view a map's geographic data and layout views to perform a broad range of advanced GIS tasks[9]. The standard interface of ArcMap is shown in figure 2.2. Maps have a page layout containing a geographic window, or a data frame, with a series of layers, legends, scalebars, North arrows, and other elements.

The toolboxes used in this framework include the Network Analyst Tools, Spatial Analyst tools, 3D Analyst Tools and Data Management Tools. The primary

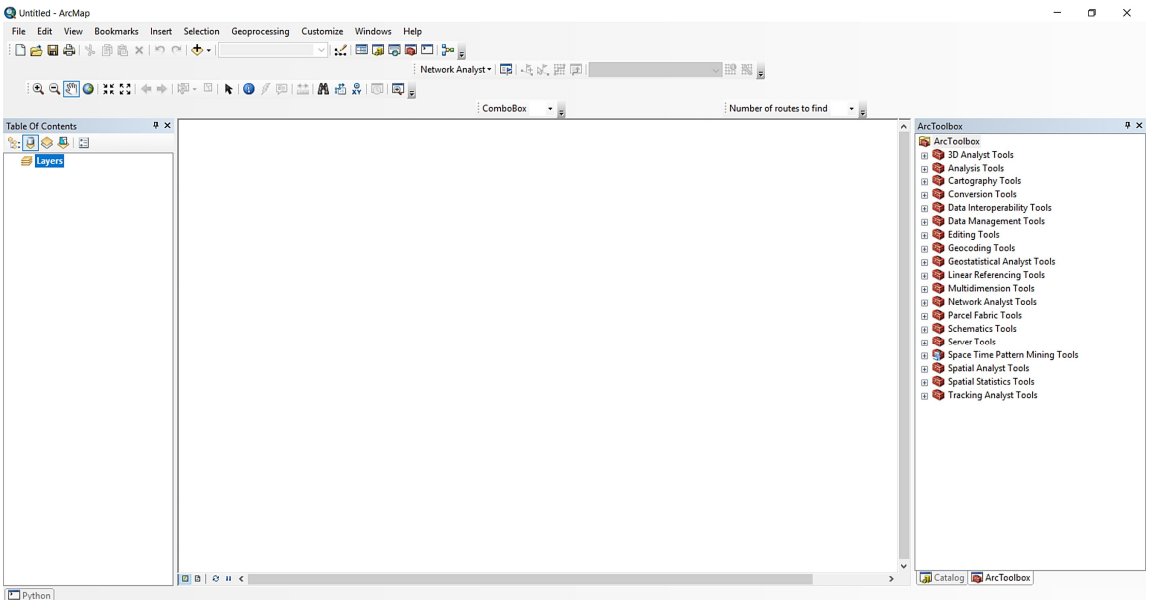


Figure 2.2: Interface of ArcMap of ArcGIS Desktop

functionality of these toolboxes is discussed in the following sections.

2.2.1 Network Analyst Toolbox

ArcMap has a special toolbox dedicated to creation and analysis of Network datasets. The toolbox, known as Network Analyst toolbox has specific input requirements and is only capable of processing Network dataset. Network datasets are made of network elements. Network elements are generated from the source features used to create the network dataset[8]. The geometry of the source features helps establish connectivity. In addition, network elements have attributes that control navigation over the network. There are three kinds of network elements:

- EdgesConnect to other elements (junctions) and are the links over which agents travel
- JunctionsConnect edges and facilitate navigation from one edge to another
- TurnsStore information that can affect movement between two or more edges

Edges and junctions form the basic structure of any network. Connectivity in a network deals with connecting edges and junctions to each other. Turns are optional elements that store information about a particular turning movement; for instance, a left turn is restricted from one particular edge to another.

2.2.2 Spatial Analyst Toolbox

The ArcGIS Spatial Analyst extension provides a rich set of spatial analysis and modeling tools for both raster (cell-based) and feature (vector) data. The capabilities of Spatial Analyst are broken down into categories or groups of related functionalities.

There are several ways to access Spatial Analyst functionality. With geoprocessing, operations in the Spatial Analyst toolbox can be performed through a Tool dialog box, Python (either at an interactive command line interface or with a script), or a Model. Traditional operations and workflows using Map Algebra can also be performed in the Python environment. There is also a Raster Calculator available for entering simple Map Algebra expressions that generate an output raster.

2.2.3 3D Analyst Toolbox

The 3D Analyst toolbox provides a collection of geoprocessing tools that enable a wide variety of analytical, data management, and data conversion operations on surface models and three-dimensional vector data.

3D Analyst tools provide the ability to create and analyze surface data represented in raster, terrain, triangulated irregular network (TIN), and LAS dataset formats. Analysis of geometric relationships and feature properties, interpolation of raster and various triangulated irregular network (TIN) models, and analysis of surface properties are only some of the numerous functions provided by the 3D Analyst tools.

2.2.4 Data Management Toolbox

The Data Management Toolbox provides a collection of toolsets that are used to develop, manage, and maintain the features within coverage and their attribute tables. These tools allow the addition of fields (items) to tables, join tables, simplify lines, and build topology. There are also tools to create new coverage, manage coverage projections, append coverage, or convert features within a coverage from one feature class to another.

2.3 Datasets

The Data supporting this framework was acquired from a range of sources. These are listed in table 2.3. The Road Networks data is further processed to give the network dataset and the DEM is used to provide elevation values for each element in the network dataset.

DATASET	SOURCE
City of Houston Roads Networks	City of Houston Geographical Information System Portal
Digital Elevation Model- 10 meters	Texas National Resources Information Systems

Table 2.1: Datasets used in framework

2.4 Study Area: City of Houston

This framework focuses on developing an intelligent evacuation routing system for the city of Houston. The study area is restricted to the road network available for the City of Houston. As the framework focuses on vehicle routing the base Map used is the World Street Map provided by ESRI as an open dataset [11]. The overall study area for this framework, marked by the network dataset is shown in figure 2.3 and a closer view of the street segments is shown in figure 2.4.

The combination of the World Street Map and the network dataset form the base layer of spatial data for the GIS routing. It is important to note that the Road dataset used to build this network covers a length of approximately 21,826 miles. However, the total road network length as reported by TomTom [20] is approximately 24,835 miles. This is a loss of about 12.11% of roads. The different categories of road classes and their distribution is as shown in table 2.4.

S. No	Road Class	Road Counts
1	Abandon	125
2	Access	13
3	Alley	83
4	Freeway	850
5	Frontage	5844
6	Highway	955
7	HOV	75
8	Local	175622
9	Major	30473
10	Minor	15
11	Private	240
12	Proposed	84
13	Ramp	3237
14	Toll way	435

Table 2.2: Frequency of each Road Class

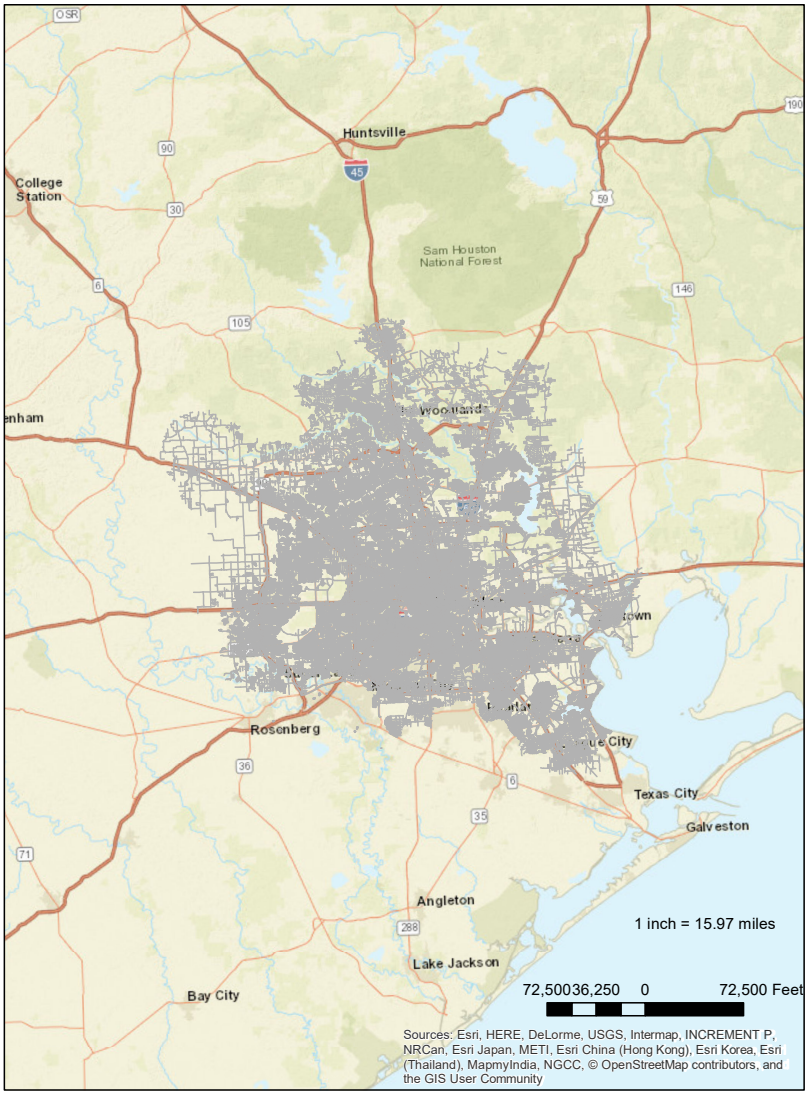
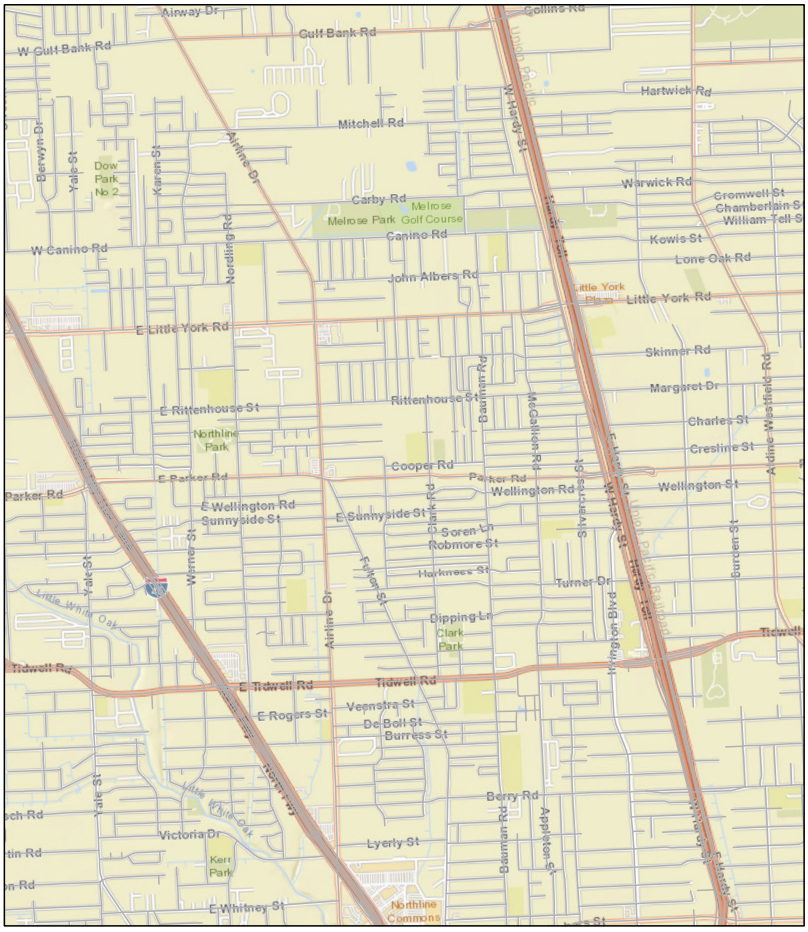


Figure 2.3: Base layer for network data for City of Houston



Service Layer Credits: Sources: Esri, HERE, DeLorme, USGS, Intermap, INCREMENT P, NRCan, Esri Japan, METI, Esri China (Hong Kong), Esri Korea, Esri (Thailand), MapmyIndia, NGCC, © OpenStreetMap contributors, and the GIS User Community

2,500 1,250 0 2,500 Feet

1 inch = 0.53 miles

Figure 2.4: Street View of a section of the Network data

Chapter 3

Related Work

Route search algorithms are a major part of applications of Artificial Intelligence. It consists of finding a sequence of actions that transforms some initial state into some desired goal state given the specifics of the environment. Planning for flood evacuation is a highly specialized environment of route planning problems. This chapter provides a detailed insight into the various approaches for path search developed throughout literature. This also involves an introduction of the different path finding algorithms with special emphasis on Dijkstra's algorithm that is the foundation to many real-world route-finding implementations. We first investigate some Artificial Intelligence (AI) planners as a prospective path planners and their drawbacks. Next, we study the development of various flood evacuation strategies proposed by various researchers and governmental organizations. A lot of these planning practices are actively in use today. This chapter also includes a section on the use of GIS technology as a medium of visualization of road networks and path finding problems.

3.1 AI Planners

This section studies two Artificial Intelligence Path Planners that were investigated during the course of development of this framework. These approaches were ultimately found to be too difficult to extend to accomplish our research goals and were therefore abandoned.

3.1.1 PRODIGY

PRODIGY is an integrated planning and learning system that can learn control rules, conduct experiments to acquire new knowledge, generate abstraction hierarchies and use analogical reasoning to recognize and exploit similarities between problems. PRODIGY uses backward chaining planners (both total order and partial order) with elements of forward chaining for a faster solution. All solutions found by PRODIGY are always considered correct [13]. The approach used by the authors for dynamic route planning includes reusing past routing cases for new plan generation. This is done by applying Case-Based Reasoning (CBR) methods developed within the framework of PRODIGY[21]. CBR is a case reuse strategy that stores derivational plans of both successful and failed decisions made by the planner. Additionally, it also holds the justifications for all these decisions. Each time a new case is encountered, the primal instinct of PRODIGY is to use CBR to reuse past decisions by reinterpretation of justifications within context of new problem. To do this, PRODIGY has a detailed mechanism for case storing and retrieval where a plan is broken into smaller cases at the intersection points to maintain constraints. The

resulting graph is called the case graph. For case retrieval, a similarity metric considering the geometric and continuous valued characteristics of a city map to generate multiple and partial cases. Each stored case is assigned an efficiency value which is a measure of how much a known case should be preferred to building a solution from scratch. Thus, the cost calculation for path traversal is equal to distance travelled for unknown regions and it is equal to times the distance travelled for known regions. The algorithm used in the paper to calculate similarity metric consists of the following features:

- Delaunay Triangulations: the principle that one is most likely to travel from one node to its adjacent node and skipping all interactions between distant vertices.
- Edge Costs: Taking advantage of the locality principle, we consider only the edge cases for triangles in the vicinity of the start and end points. Further, domain knowledge (using CBR) helps in determining the simplest routes.
- Shortest Paths: the final step in the similarity metric involves traversing the triangulation as a graph using Dijkstra's shortest path algorithm to find the optimum route[14]. Dijkstra's shortest path algorithm involves finding local minima paths in the triangulations to give a global minimum path for case optimization.

Once a set of cases has been retrieved the plan is executed. An important part of plan execution is learning. Each time a plan is being executed, it must be broken

into smaller cases. Execution of each small case must be followed by evaluation and learning from the environment depending on the action taken. Doing this is an integral part of making the planner sensitive to the dynamic nature of the environment. The following metrices are adjusted in each case traversal to reflect the changes in the environment:

- Failure identification: When a time bound success is not achieved, the system starts to identify the category and severity of the failure to record them. This are useful for further traversals of the same case and are integral control rules in case of replanning, if it is ever needed.
- Adjustment of values: To truly build the knowledge of a planner, experience of each case and its reiterations are reflected in the form of alterations to the values. These alterations are noted along with their justifications so as to help in obtaining better similarity metrices for further newer scenarios.

3.2 Shortest Path Problem

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. The problem of finding the shortest path between two intersections on a road map (the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment) may be modeled by a special case of the shortest path problem in graphs [24]. To be able

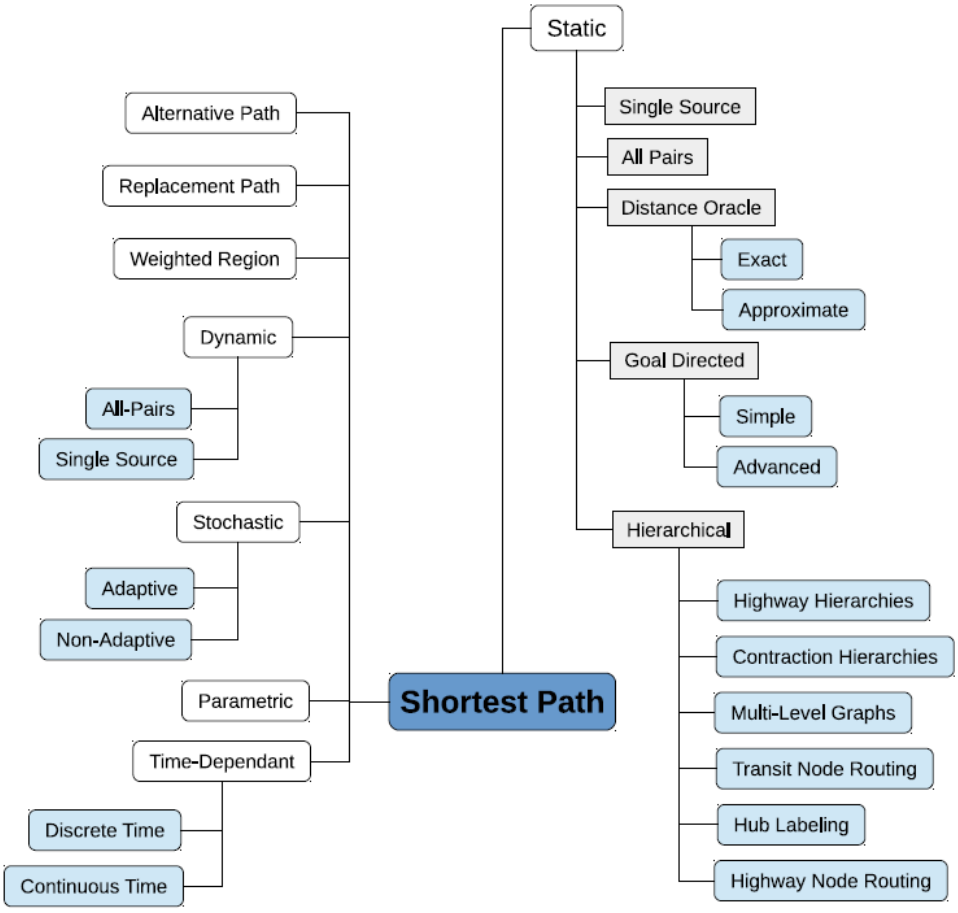


Figure 3.1: Taxonomy of Shortest Path algorithms [17]

to study the shortest path problem in its complete depth it is important to have a certain organization or hierarchy of the proposed flavors of solutions throughout history. This has efficiently been represented as a taxonomy in Figure 3.1 proposed by Madkour et al.

The static branch in Figure 3.1 lists algorithms that operate over graphs with fixed weights for each edge. The weights can denote distance, travel time, cost, or any other weighting criteria. Given that the weights are fixed, some static algorithms perform precomputations over the graph. Goal-directed algorithms optimize in terms

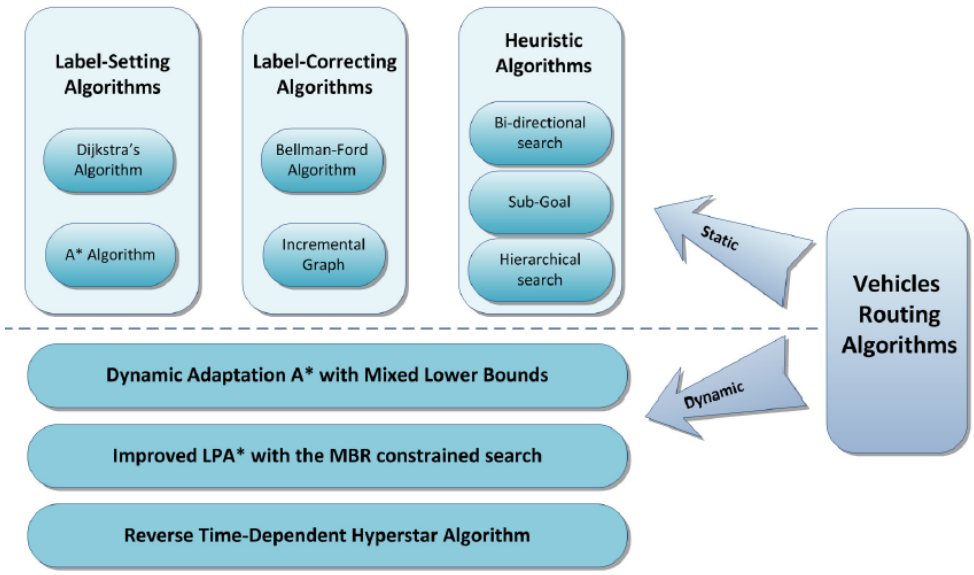


Figure 3.2: Classification of vehicle routing algorithms [22]

of distance or time toward the target solution. The dynamic branch in Figure 3.1 lists algorithms that process update or query operations on a graph over time. The update operation can insert or delete edges from the graph or update the edge weights. The query operation computes the distance between source and destination vertices. Time-dependent algorithms target graphs that change over time in a predictable fashion [17].

However, it is important to note that shortest path algorithms are seldom used in their original form for solving routing problems across a street network. Street Networks are graphs with millions of nodes and edges. Further, each segment is annotated with a lot of geographical, spatial, and other features to successfully conduct navigation. A popular classification of Vehicle routing algorithms proposed by Wang et al. is shown in figure 3.2.

A general classification of the existing vehicles routing algorithms is shown in

Figure 3.1 in which we distinguish two main classes of algorithms: static and dynamic. The static algorithms include, amongst others, Dijkstra's Algorithm (DA) [5], and its improved version A* [19] in which the Euclidean Distance is introduced as the lower bound to ensure it would never overestimate the real travel distance between the origin and the destination nodes. The routing algorithms of the second class (i.e., dynamic) consists of Dynamic A*, improved LPA*, etc. These algorithms are more practical in real transportation scenario as the weight of each link in the graph representing the road network is changing over time to reflect real time traffic congestion levels and any road bloackage incidents [22].

Several modifications of these popular algorithms were proposed for route finding algorithms based on specifications of environment. Most of these algorithmic variations provide suboptimal solutions however each is designed to solve a unique problem.

Felner et al. developed the Physical-A*(PHA*) routing algorithm [12]. PHA* expands all the mandatory nodes that A* would expand and returns the shortest path between the two points. However, the complexity of the algorithm is measured by the traveling effort of the moving agent [12]. PHA* is designed to minimize the traveling effort of the agent by intelligently choosing the next assignment of the traveling agent. In this manner PHA* enables discovery of paths in unknown to partially known environments.

Another popular algorithm proposed for route navigation is based on randomization principles. Chakraborty proposed a route selection for car navigation using

the Genetic Algorithm(GA) [3]. A GA based algorithm is used to find out simultaneously several alternate routes depending on different criterion according to drivers choice such as shortest path by distance, path which contains minimum number of turns, path passing through mountains or by the side of a river etc.

3.3 GIS for Path Planning

GIS is an integral part of visualization of transportation and road networks. GIS and transport research have always been interrelated. Thus, it is hard to ultimately decide whether transport modeling is an application domain of GIS or spatial capabilities are incorporated in transport models[16]. Examples exist in both domains and current transport modeling software products increasingly provide integrated GIS capabilities. GIS are capable environments for the capturing, management, analysis, and visualization of spatial data. They allow for an integration of various data sources into a scalable, dynamic, and adaptable geospatial framework as mentioned in figure 3.3. Through models, simulations, and analyses, each with an explicit consideration of the spatial nature of transport, new information can be generated. Besides, GIS also facilitates information visualization which serves as a communication platform with feedback loops to the data integration and the settings of models, simulations, and analyses[16].

In their study Chandio et al. created an integrated multi-criteria decision analysis (MCDA) and Least Cost Path Analysis (LCPA) approach to determine the best route for a road given the topography and function relating slope, land use and cost using

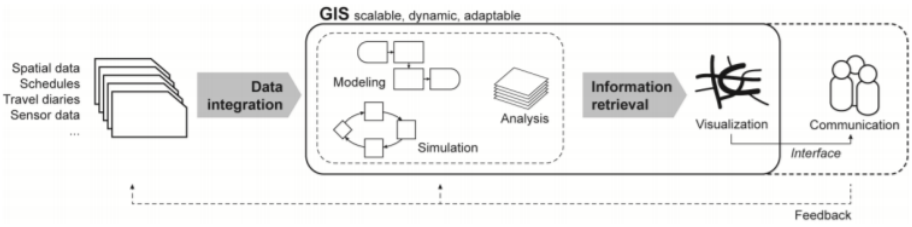


Figure 3.3: Capabilities of a GIS environment for transport modelling

the GIS. The proposed methodology is tested on Kulai Senai Town, Johor Bahru, Malaysia [4].

Chapter 4

Methodology

In this chapter we first introduce the design of the proposed framework. The primary goal of the propose framework is to establish multiple suboptimal paths between two given locations on a map. Furthermore, the provided paths are annotated with several attributes of those routes which help in evaluating their quality and safety.This chapter starts by describing the problem statement, the environment and the different variables involved. This is followed by a detailed description of the spatial datasets used in this thesis. We then proceed to enhancing the network dataset by factoring in the land elevation of each road segment. This forms the integral preprocessing step for implementing the algorithms proposed in this thesis. The flowchart in figure 4.1 depicts all the steps that form the framework. Once the network dataset is engineered and the elevation fishnet is created, we find the most optimal route between required locations using a variation of Dijkstra's shortest path algorithm. Once this route is found we use either one of the two proposed variations of the

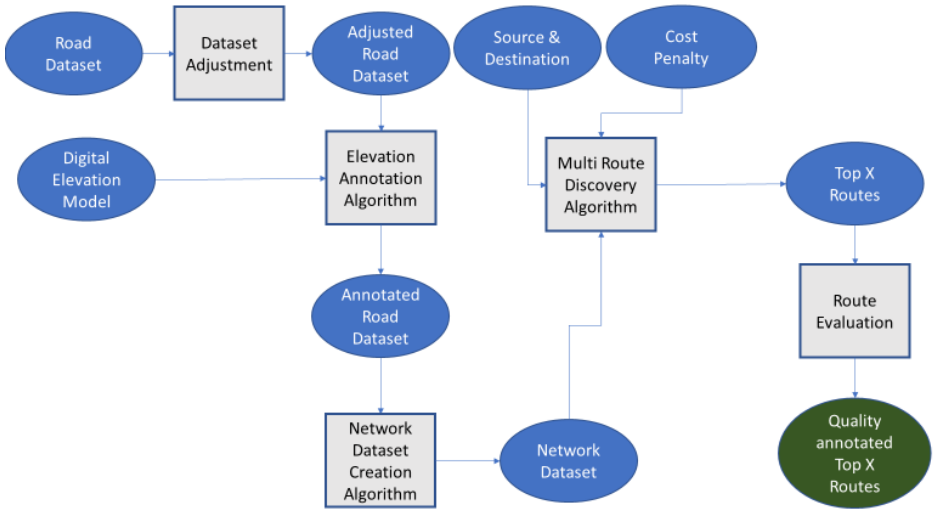


Figure 4.1: Architecture of proposed framework

algorithm for finding alternative paths.

Once alternative routes are obtained, a detailed evaluation of all the associated parameters is conducted. A tradeoff is established between competitive variables and the top 3 paths are discovered.

4.1 Design of proposed framework

The environment for the proposed framework is the City of Houston. A Road dataset consisting of all the roads of the City of Houston, obtained from the City of Houston Geographical Information System (COHGIS) is used to build the Network dataset. This complex network structure is represented in the form of a graph where a vertex denotes a street intersection and an edge denotes a road segment. Each edge on the graph is associated with a tuple {x, y, z} where x is the Length of a street segment,

y is the Maximum Speed Limit of a street segment and z is the elevation of the street segment

The value of time taken to travel a street segment (t) is calculated using the formula mentioned in equation 4.1.

$$t = street \quad segment \quad length/maximum \quad speed \quad limit \tag{4.1}$$

The algorithm used for graph traversal is a modified version of Dijkstra's algorithm which is described in detail later in this chapter.

4.2 Network Dataset Creation

The creation of a Network Dataset is an integral part of the framework. Network datasets are well suited to model transportation networks. They are created from source features, which can include simple features (lines and points) and turns and store the connectivity of the source features. The list of features that are integral to building a network dataset are as mentioned in table 4.2.

When building the final network dataset, it is integral that all source features participating in the network creation have acceptable values. It was observed that the maximum speed limit feature for some roads was set to 0 (due to incomplete data).The formula for finding the travel time is shown in equation 4.2. This lead to failure in creation of all these roads as their travel time was calculated as ∞.

Number	Road Network Feature
1	StarMap_ID
2	Street Type
3	Name/ Alias
4	Left/ Right Address
5	Road Class
6	One ways
7	Number of Lanes
8	Left/ Right subdivision
9	Left/ Right City
10	Left/Right County
11	Left/Right State
12	Left/Right Postal
13	Left/Right ZIP
14	Left/ Right ESN (Esri Street Number)

Table 4.1: Road Network essential features

$$Traveltime = Road \quad segment \quad length/maximum \quad speed \quad limit = distance/0 = \infty \tag{4.2}$$

To correct this error in network dataset creation, all roads with maximum speed limit mentioned as 0 were assigned a speed limit of 30 miles\hr. Once the data integrity issue is resolved, a network dataset is created successfully using the ArcMap Network Dataset Wizard.

The road network so build consists of 218051 road segments. However, there are several roads which are classified as either Abandoned or Proposed roads. These are 209 road segments were removed for the final network dataset creation.

4.3 Elevation Modelling

Elevation for the city of Houston is available in the form of a Digital Elevation Map (DEM). DEM is the representation of continuous elevation values over a topographic surface by a regular array of z-values(indicating the point elevation values), referenced to a common vertical datum. DEMs are typically used to represent the bare-earth terrain, void of vegetation and man made features. The DEM used in this framework is produced by the USGS at a 10-meter grid size from contour data. However, in its raw form, the obtained DEM data is very dense. This dense nature of the data leads to a high data volume. As the proposed framework runs of a single instance of ArcGIS, handling such volumes of data is impossible for the framework. This calls for a need of reducing the elevation data such that the framework can handle it. This is done by converting the USGS DEM data into sparse grid data. We create a grid (popularly referred to as a fishnet in ArcGIS) of 500*500 across the city map. The algorithm used to transform DEM to elevation data is described in figure 4.2. The algorithm takes the input of a DEM. It then creates a 500 * 500 grids fishnet that has the same spatial expanse as the DEM. The fishnet so created is annotated with its spatial attributes including its coordinates and geocoded locations. Now, the value of elevation for each intersection on the grid is obtained. Once the elevation values for all cells in a grid are extracted, they are annotated to each street segment. The annotation is done by finding all the grids that interest a road segment and assigning it the minimum elevation of all these grids. This is the final step in the annotation algorithm.

ALGORITHM 1. ELEVATION MODELLING ALGORITHM

```
1. Procedure  GridBasedElevation(DEM)
2.        GridFormation(Road_dataset_boundary)
3.                num_fishnet_rows = 500
4.           num_fishnet_cols = 500
5.        row_size= Road_dataset_boundary/num_fishnet_rows
6.        col_size = Road_dataset_boundary/num_fishnet_cols
7.           fishnet_coord_extent ← Coordinate System environment variable
8.           Create fishnet with geometry type polygon to assign elevation value to each grid cell in the
fishnet
9.        Grid = Create_Fishnet(num_fishnet_rows,num_fishnet_cols, fishnet_coord_extent)
10.          return (Grid)
11.       End GridFormation
12.       Clip DEM to the extent of the grid formed
13.       GridValueExtraction(Grid,Clipped DEM)
15.       PopulatedGrid= Assign elevation value to each grid using value at cell center
16.          PopulatedGrid = Assign Coordinates(PopulatedGrid)
17.      End GridValueExtraction
18.       AssignElevationToStreet(Network dataset,Grid)
19.       While(Network dataset = True)
20.               Elevation(road segment) = minimum(elevation of all intersecting grids)
21.       End AssignElevationToStreet
```

Figure 4.2: Elevation Annotation Algorithm

4.4 Dijkstra's Algorithm

The proposed approach in this thesis is independent of the underlying routing algorithm used for shortest path finding. The classic approach to routing in a road network with barriers is using the Dijkstra's algorithm [4]. This is used as the base algorithm for all routing operations and is described in detail in figure 4.3 . The Dijkstra's algorithm takes a road network, elevation threshold, start node and goal node as input and returns an optimal path to minimize a certain attribute. The path so returned by Dijkstra's algorithm is the shortest path between the start and goal state given a road network. The algorithm basically begins at the source location. It then tries to find the closest vertex in the direction of the goal state. The direction of the goal state is determined by the euclidean distance between the current state

ALGORITHM 2. DIJKSTRA'S ALGORITHM

```
1. Procedure findPath(Road Network dataset, start vertex,end vertex, minimizing attribute)
2.      distance[source] ← 0                                    (distance to source vertex is zero)
3.              for all v ε V – {s}
4.              do dist[v] ← ∞                                  (set distances of all other vertices to ∞)
5.        while Q ≠ φ
6.      do      u ← minimumDistance(Q,dist)          (select element of Q with minimum distance)
7.                      S ← S U {u}                             (add u to queue of visited nodes)
8.                      for all v ε neighbors[u]
9.                      do      if dist[v] > dist [u] + w(u,v)          (if new shortest path found)
10.                             then d[v] ← d[u] + w(u,v)               (set new value of shortest path)
11. Path = pointToEdge(S)                                       (convert all points in S to edges
in road network)
12. Return Path
```

Figure 4.3: Elevation Modelling Algorithm

and the goal state. This continues for all subsequent vertices until the goal state is obtained.

4.5 Alternative Route finding Algorithms

The proposed approach to finding alternative routes is based on the key idea of penalizing the found solution by increasing the cost of one segment or the whole path, forcing the search algorithm to find a different, alternative solution. The black box for this proposed algorithm is described in figure 4.4 .

We propose two variations of the algorithm. The first algorithm described in algorithm 3, involves finding cost optimal path and then penalizing the whole path to find subsequent alternative routes. This is integral in replicating the scenario where all road segments of the optimal path have an extremely low elevation relative to the flood levels. The second proposed variation described in algorithm 4, involves finding

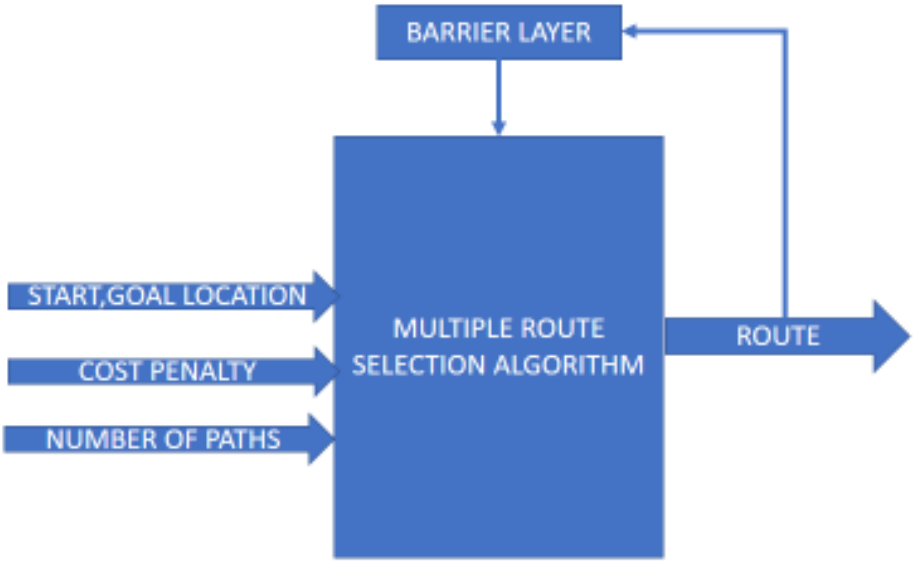


Figure 4.4: Proposed approach for multiple route selection

an optimal path and subsequent recalculations by penalizing one random road segment that is a part of the shortest path algorithm solution. This is particularly useful in rerouting from a small section of the network dataset that may be inaccessible due to localized flooding, temporary construction, or localized traffic congestion.

4.5.1 Path Penalty Algorithm

The path penalty algorithm can be used to generate alternative global solutions across a given street network. To do this is ArcGIS we have to load the network dataset and initiate a new route analysis layer. These include a minimum of two stops, or the start and end points. Although this is the only required feature for routing, additional features such as Barriers (Point, Line or Polygon barriers) are also supported for route analysis. We make use of python scripting to invoke the

ALGORITHM 3. PATH PENALTY ALGORITHM

```
1. Procedure  Path_Penalty(Start location, goal location, route layer, barriers layer)
2.       Kpath(Map document)
3.       Clear(Barriers Layer) #cleaning all existing barriers from previous runs
4.       Clear(Route Layer) #cleaning all existing routes from previous runs
5.       First_Route = Route solver(start location, destination, Map document) #finding first route
6.       Route Layer + = First_Route #add first route to solutions
7.       while(i <= k - 1) #find next k-1 routes
8.                cost_penalty = x #establish path repitition penalty
9.                createNewBarrier(Route Layer, cost_penalty) #create new barrier layer
10.               loadBarriers(Barriers Layer)
11.               new Route +=  Route solver(start location, destination, Map document, barriers layer)
                  #find next route with changed variables
12.               Route layer +=new Route #add new route to solutions
13.      return Route layer
14.     End Kpath
```

Figure 4.5: Path Penalty Algorithm

route analysis layer repeatedly to get top k routes between the two locations. In the path penalty algorithm each new route becomes a line barrier for all subsequent routes. This means each time a new route is to be created, the cost of using all previous routes is increased by a certain penalty. This forces the routing algorithm to find different routes across the street network.Figure 4.5 gives the pseudo code of the path penalty algorithm. We first begin by finding the first route. This is followed by creating a line barrier of the route so obtained as a cost scaled penalty. Then we find the next route again. This procedure is repeated until the number of required roots is found. This algorithm helps us find extremely disjoint k routes. The algorithm requires a start and goal location in addition to the cost penalty that is applied to get subsequent routes. Additionally, the algorithm requires the route layer where the results of finding each route is stored and a barrier layer that stores the cost barrier for each time routing is performed.

4.5.2 Randomized Segment Penalty Algorithm

The randomized segment penalty algorithm is a variation of the path penalty algorithm. Here, instead of penalizing the whole of previously obtained routes, we penalize only a single road segment. This specialized case has been developed to cater for localized modifications in routing. It is especially helpful to replicate the real-life scenarios of localized flooding, temporary closures, etc. The road segment chosen to incur penalty is selected randomly by a method known as Sampling. Figure 4.6 gives the pseudo code for the randomized segment penalty algorithm. This algorithm begins by finding a route. This route is then divided into the respective road segments. One road segment is randomly picked and loaded as a cost penalty line barrier and the search resumes for the modified network.This procedure is repeated until the number of required roots is found. This algorithm produces high level of duplication in the dataset of routes found.

4.6 Analysis of top k routes

Once a certain number of alternative routes has been obtained, we perform an analysis of the road segments that are a part of the routes. This begins with breaking down the routes into individual road segments by splitting them at the road intersections. This is followed by a spatial join of the obtained segments with elements of the road dataset in order to recover network properties. Once all the properties/ features of the road segments are obtained we perform basic arithmetic operations to obtain various parameters including route length, true travel time of route, width of

ALGORITHM 4. RANDOMIZED SEGMENT PENALTY ALGORITHM

```
1. Procedure  Segment_Penalty(Start location, goal location, route layer, barriers layer)
2.       Kpath(Map document)
3.       Clear(Barriers Layer) #cleaning all existing barriers from previous runs
4.       Clear(Route Layer) #cleaning all existing routes from previous runs
5.       First_Route = Route solver(start location, destination, Map document) #finding first route
6.       Route Layer + = First_Route #add first route to solutions
7.       while(i <= k - 1) #find next k-1 routes
8.                cost_penalty = x #establish path repitition penalty
9.                Lines_in_path = ConvertPolylinetoLine(Route Layer)
                  #convert all route polylines to corresponding road segments.
10.               Penatly_segment = Sampling(Lines_in_path) #randomly choose one road segment
11.               createNewBarrier(Penalty_segment, cost_penalty)
                  #create cost penalty barrier across randomly chosen road segment
12.               loadBarriers(Barriers Layer)
13.               new Route +=  Route solver(start location, destination, Map document, barriers layer)
                  #find next route with changed variables
14.               Route layer +=new Route #add new route to solutions
15.      return Route layer
16.    End Kpath
```

Figure 4.6: Randomized Segment Penalty Algorithm

road segments, percentage of duplicate roads, percentage of congested roads involved in routing, minimum and average elevation and minimum and average slope. All of these factors are used in assessing the quality and safety of the routes obtained.

Chapter 5

Implementation

The implementation of this thesis includes the usage of ArcGIS and its tools for various processing, visualizations, and analysis. This chapter provides the implementation details of the software framework that was developed in this thesis. The first implementation task is the preparation of base layer. This includes creation of the road network dataset from the roads dataset and limiting it to the boundary of City of Houston. Once the base layer of the framework is defined all other datasets used in the framework are created using a special Geoprocessing operation called Clip. This helps in defining spatial bounds to create an Area of Interest (AOI) for all features under consideration. Clip is performed for reducing the storage load of the project which leads to faster processing. Figure 5.1illustrates the operation of Clip to produce a spatially limited output.

Both the path penalty and randomized segment penalty routing algorithms proposed in the framework use certain inbuilt capabilities of ArcGIS for routing. To

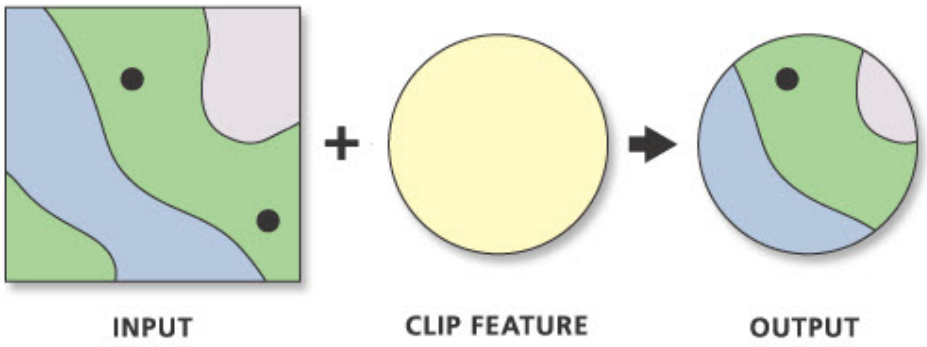


Figure 5.1: Clip operation performed on features

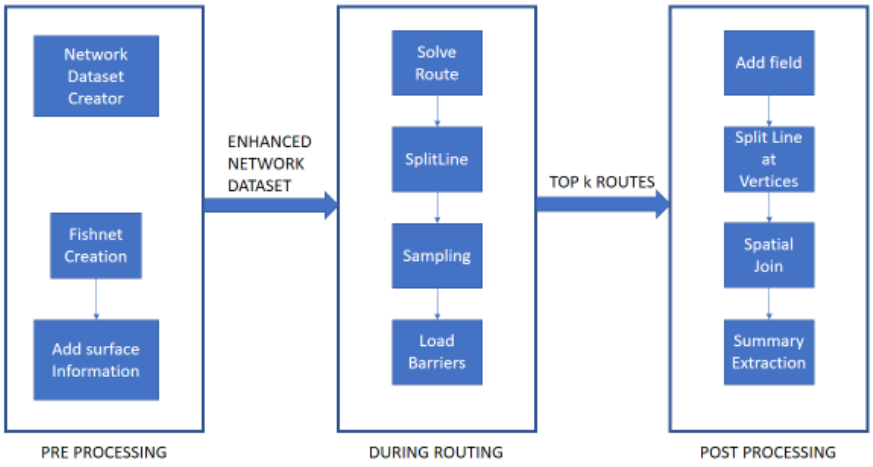


Figure 5.2: ArcGIS tools used throughout the framework

alter their behavior to suit the study, ordinary shortest path routing between two points is performed in conjunction with several intermediate operations. Figure 5.2 summarizes all the tools used during preprocessing, during the actual multi path routing and in the post processing stages.

The following sections describe the usage of the various tools.

5.1 Pre Processing

The tools used during preprocessing include the network dataset creator. This is also called the Network Dataset Wizard. The New Network Dataset wizard gives the options of naming the network dataset, identifying the network sources, setting up the connectivity, identifying elevation data (if necessary), specifying turn sources (if necessary), defining attributes (such as costs, descriptors, restrictions, and hierarchy), and setting up the direction reporting specifications. When you create a network dataset or edit an existing network dataset, it must be built. Building is a process of creating network elements, establishing connectivity, and assigning values to the defined attributes.

The Create Fishnet tool creates a fishnet of rectangular cells.This tool is used to reduce the elevation data such that the framework can handle it. This is done by converting the USGS DEM data into sparse grid data. The output can be polyline or polygon features. One can choose either the number of grids, or the grid size while creation. One also has the option to specify the extent of the fishnet using another dataset. Once the fishnet is created it must be annotated with the respective elevation data from the available DEM. This is done using the Add Surface Information tool which derives the spatial information of given set of features from a surface.

5.2 During Routing

During the search for multiple routes, several intermediate operations are involved. The first operation is finding a route between two points. This is done using the Solve class of the Route Analysis tool. Selection of a lot of parameters is involved when using the routing tool. These include any barriers, any date and time restrictions and type of shape of output route. The Split Line at vertices tool is used to create a feature class containing lines that are generated by splitting input lines or polygon boundaries at their vertices [3]. This is depicted in figure 5.3. This tool is used to break the output route polyline into the respective road segments. The Sampling tool is used to randomly select one segment from all the road segments involved in routing. This is subsequently loaded as a randomly obtained barrier segment.

5.3 Post Processing

Once the top k routes are obtained, the post processing or analysis of the routes is performed. If any operation is performed on the routes obtained they tend to lose their identity as a unique route. This in turn leads to structural disruption of the result. Therefore in this stage, we first assign a unique ID to each route to preserve the structure of the dataset. This is done by the Add Field operation. To perform any analysis over the road segments that are a part of all the top k routes, we split the k routes into road segments using the Split Line at Vertices tool described in section 5.2. Once the split is performed, we need to find the necessary features of

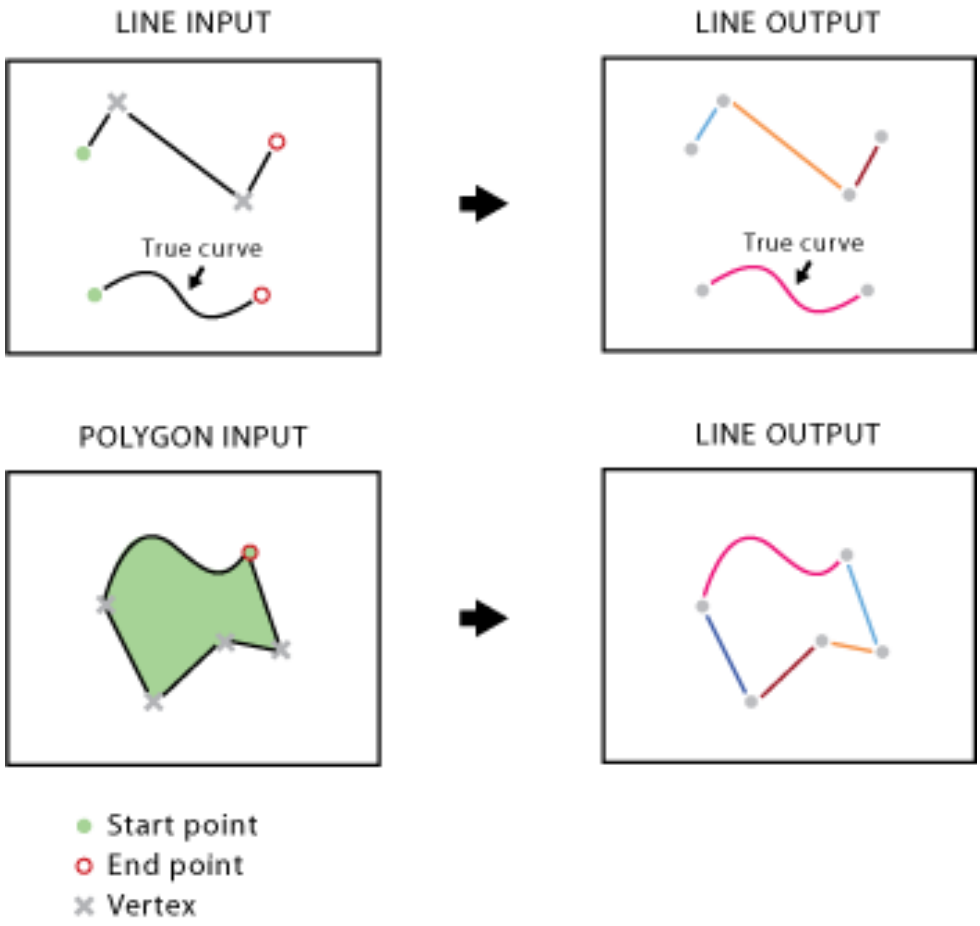


Figure 5.3: Demonstration of Split Line Tool

these roads segments from the road dataset. This is done using the "Spatial Join" tool. Joins attributes from one feature to another based on the spatial relationship. The target features and the joined attributes from the join features are written to the output feature class. The final step of the framework is producing a summary of the various parameters of the 20 routes, this is done using the Summary Statistics tool. The operations available with the tool include Sum, Mean, Maximum, Minimum, Range, Standard Deviation, Count, First, and Last. The Median operation is not available.

Chapter 6

Experimental Evaluation

In this section, we evaluate our framework by performing three different experiments. All these experiments are performed for the Network dataset developed for the city of Houston. Figure 6.1 visualizes of the road network dataset for the City of Houston. The first section of this chapter focuses on establishing the evaluation parameters used in testing the routes found. Standard route planning algorithms usually generate a minimum cost solution based on a predetermined cost function. Unfortunately, such a solution may not represent the desirable routes for evacuation during or post disasters. Several parameters are involved for measuring the desirability of routes in the events of evacuations. These range from the length of routes, the time taken to travel it, the safety of the routes depending on the flooding levels, the type of roads selected for routing and the slope of the road segments. Depending on the intensity and stage of flooding it is important to prioritize the aspect of safety over all other parameters. The evaluation parameters used in the experiments are used as a post

processing step to give additional evaluation data for the routes found.

Section 6.2 describes all the experiments performed to test the algorithms developed in this thesis. A benchmark of 5 start locations and a single goal location is used to test the proposed framework. The first experiment involves finding a single route for all the start locations in this randomly generated dataset. All of this together form the baseline measurements of all the experiments that follow. The second experiment involves finding top three routes between each start location and the goal locations using the two different algorithms developed during this thesis. A comparative analysis of the results obtained by the two algorithms is performed using various evaluation measures. The final experiment involves testing the multi routing system for different elevation thresholds to find the effect of increasing flood levels on the routes obtained by the two versions of the proposed algorithms.

6.1 Evaluation Parameters

The following section provides a detailed description of all the evaluation parameters used to assess the quality of routes.

- Route Length: The most common parameter to assess the quality of a route is its length. In most scenarios it is preferable that the route with minimum length is selected. However, in cases of evacuation this parameter can be compromised in exchange of the safety of the road.
- Time taken to travel route: The time taken to travel across a route depends on

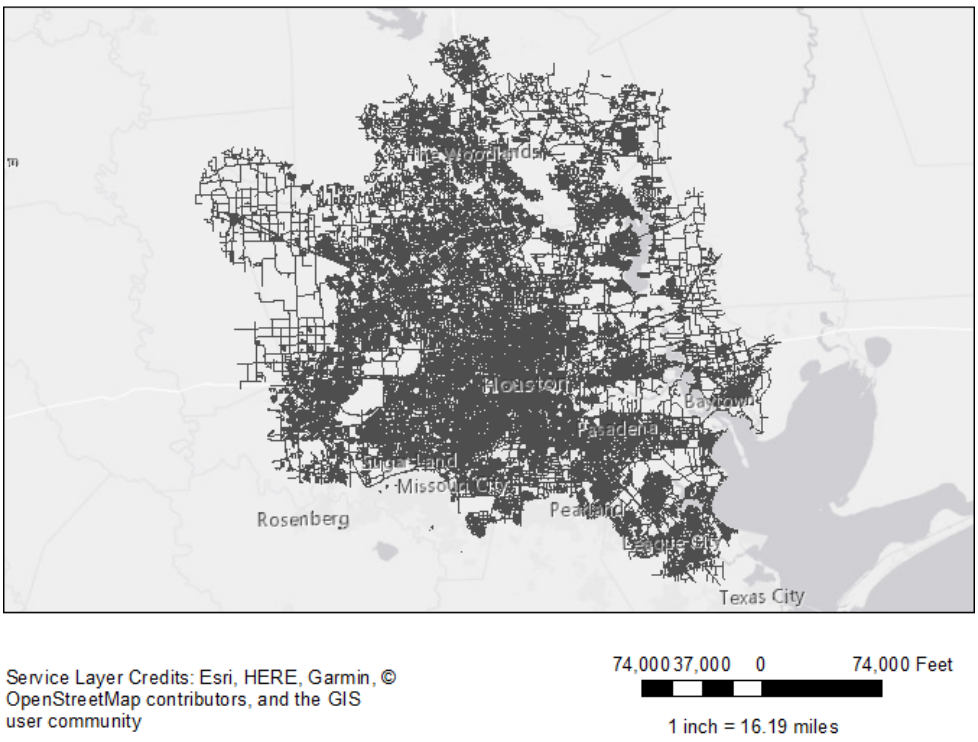


Figure 6.1: Road Network for the city of Houston

the route length and the penalties incurred by travelling certain road segments throughout the road network. The set of routes obtained are always ranked by their travel times. The combination of route length and the time taken to traverse it forms the foundation of finding multiple routes across two locations. Equation 6.1 is used to calculate the travel times of different routes.

$$Traveltime = RouteLength/PermissibleMaximumSpeedLimit \tag{6.1}$$

Further the penalties for travel time set by the proposed algorithms are calculated by increasing the travel time by a constant factor. Equation 6.2 is used to calculate these penalties.

$$PenalizedTraveltime = Originaltraveltime * penaltyconstant \quad (6.2)$$

- Quality of Routes: The quality of routes is determined by evaluation of the various attributes of the road segments involved in the routing.
  - Number of lanes of road segments: The number of lanes of road segments involved enable in judging which alternate route to choose for routing. Usually the widest routes must be selected to avoid congestion during navigation.
  - Percentage of duplicate roads: The percentage of duplicate routes is an important parameter to judge the quality of routes produced by our algorithm.
- Safety of Routes: The safety of routes is determined by evaluating its road segments for their elevation and maximum speed limits. This is particularly useful for navigation during the event of road flooding when the levels of water across different streets is constantly changing.
  - Elevation of road segments: The safety of road segments selected for a route is decided by the percentage of road segments that are significantly elevated as compared to the current flood levels. This parameter is particularly important for evacuation during the flooding.
  - Maximum Speed Limits of road segments: The maximum speed limits(MSL) of road segments help in evaluation of rad safety. It is known that wider road segments have a higher speed limit, thus it is helpful to

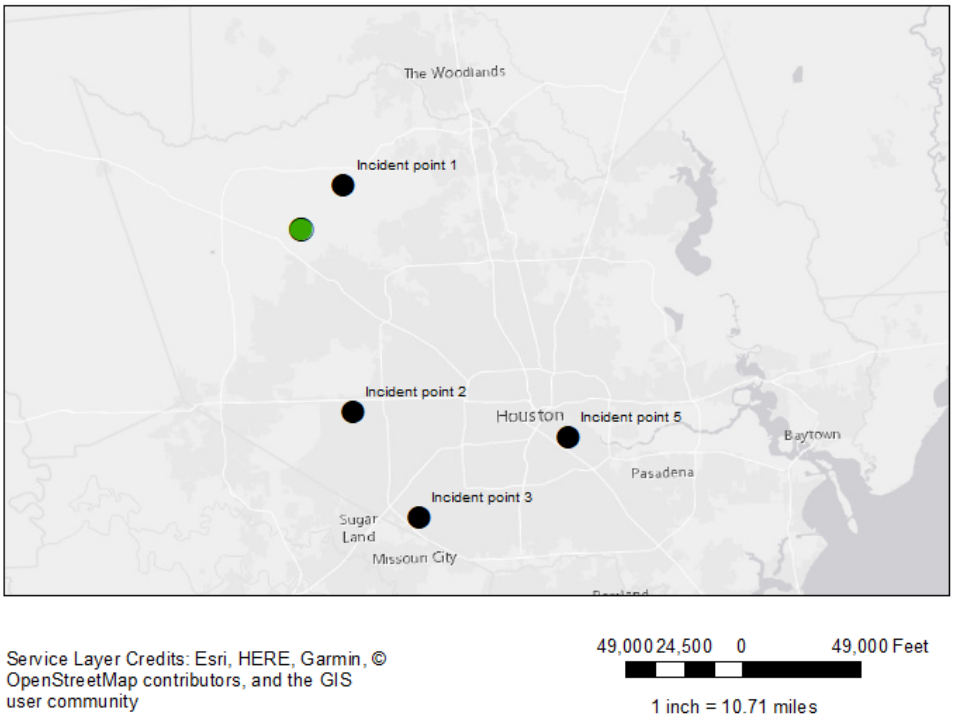


Figure 6.2: Randomly generated start and end locations

know the average MSL of the different alternative routes available during evacuations.

6.2 Experimental Evaluation

The following section presents the results of the various experiments performed using the developed framework. The experiments primarily focus on obtaining routes. However, after routes are created various parameters of the routes obtained are evaluated relative to each other.

Start ID	Goal ID	Name of Route	Total Time (min)	Total Length (miles)	Minimum Elevation(feet)	Mean Elevation(feet)
1	1	Incident point 1 - Goal	10.62	6.02	43.86	44.01
2	1	Incident point 2 - Goal	23.10	20.74	25.93	26.17
3	1	Incident point 3 - Goal	30.69	29.14	22.12	23.01
4	1	Incident point 4 - Goal	47.37	34.23	43.86	44.01
5	1	Incident point 5 - Goal	34.30	32.38	10.56	10.65

Table 6.1: Dijkstra's Algorithm Routing Results

6.2.1 Routing with Dijkstra's Algorithm

A dataset of 5 points is randomly generated. This is done by using the Sampling toolset from the Data Management Toolbox along the road edges of the road network dataset. Figure 6.2 shows the points so generated as black circles. These locations depict the start points for the experiment. Additionally, a single point is generated to act as the goal state. It is represented in figure 6.2 as a green circle.

The basic Dijkstra's routing algorithm between each of the start locations and the goal location yields the results listed in table 6.2. Figure 6.3 visualizes of all the routes on the map of the City of Houston. It is important to note that these are the baseline routing measurements when there are no dynamic variables involved affecting the route finding.

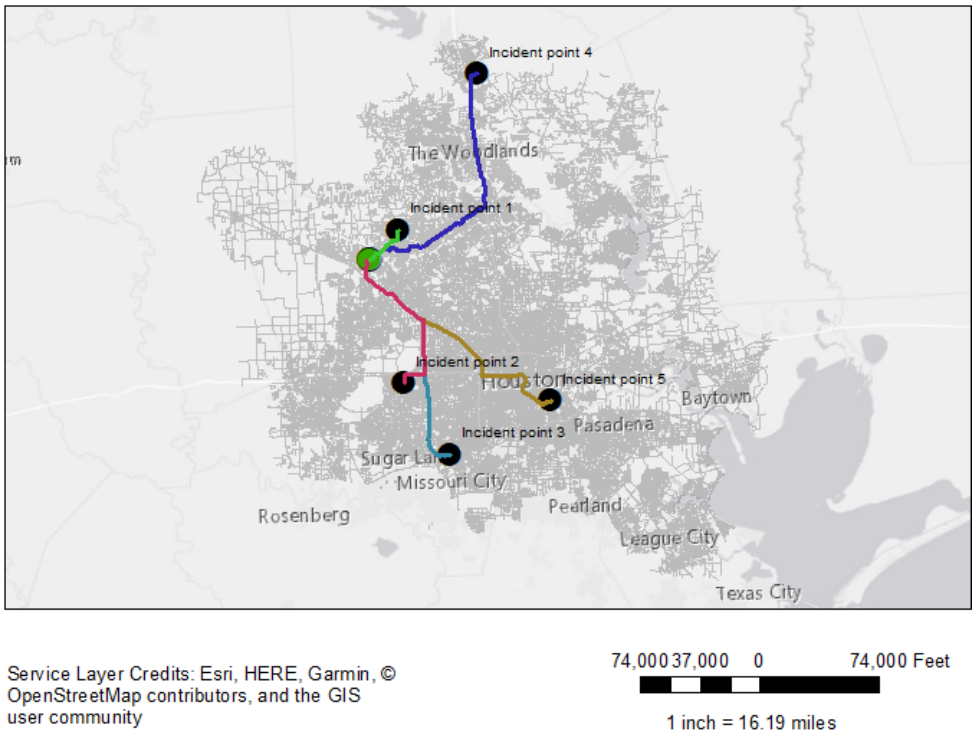


Figure 6.3: Routes obtained by Dijkstra's Algorithm

6.2.2 Penalty Algorithms

In this experiment we compare the two proposed algorithms based on various evaluation measures. Several parameters are involved for measuring the desirability of routes in the events of evacuations. These range from the length of routes, the time taken to travel it, the safety of the routes depending on the flooding levels, the type of roads selected for routing and the slope of the road segments. Depending on the intensity and stage of flooding it is important to prioritize the aspect of safety over all other parameters. The evaluation parameters used in the experiments are used as a post processing step to make an intelligent choice of route from all the routes available between two points.

Start ID	Goal ID	Name of Route	Total Time (min)	Total Length (miles)	Minimum Elevation(feet)	Mean Elevation(feet)
1	1	Incident point 1 - Goal	15.14	12.12	22.1	29.6
2	1	Incident point 2 - Goal	34.7	26.5	15.12	45.8
3	1	Incident point 3 - Goal	52.8	34.7	8.94	28.02
4	1	Incident point 4 - Goal	62.61	43.11	26.4	44.2
5	1	Incident point 5 - Goal	45.43	39	6.35	15.72

Table 6.2: Averages of PPA results

Start ID	Goal ID	Speed Limit (miles/hr)	Percentage of shared road segments
1	1	32.4	68%
2	1	41.9	52%
3	1	43.21	57%
4	1	49	54%
5	1	48.3	48%

Table 6.3: Averages of PPA Evaluation Measures

Route number	X=1.1	X=1.3	X=1.5	X=1.7	X=1.9
1	13.5	13.56	13.56	13.56	13.56
2	13.5	13.56	13.56	13.56	11.97
3	13.5	13.56	2.90	2.90	-0.12
4	13.5	13.56	2.90	2.90	14.03
5	13.5	2.90	13.55	13.55	-0.17
6	13.5	2.90	12.64	12.64	17.93
7	2.8	12.64	2.99	2.99	2.90
8	13.5	2.90	13.56	12.64	11.97
9	13.5	13.55	13.56	2.90	7.32
10	2.8	12.64	2.90	11.97	9.67
11	13.5	2.90	11.97	-0.12	17.71
12	2.8	13.55	13.55	13.56	13.56
13	13.5	2.99	-0.12	2.90	13.55
14	2.8	11.97	14.98	-0.17	2.90
15	13.5	-0.12	2.90	14.03	2.90
16	12.6	13.56	-0.17	17.93	12.64
17	2.8	2.90	10.94	9.67	2.99
18	2.8	13.55	7.32	7.32	13.55
19	12.6	2.99	2.90	11.97	2.90
20	13.5	9.67	2.99	2.90	12.64

Table 6.4: Minimum elevation(in feet) for 20 routes using PPA between started ID 3 and goal at different cost penalties marked by X

6.2.2.1 Path Penalty Algorithm

The path penalty algorithm(or PPA) focuses on obtaining highly disjoint routes. This is done by obtaining the least cost path and then penalizing the whole path to obtain the next path. We test the current framework for 5 different cost penalties including 1.1, 1.3, 1.5, 1.7, 1.9. This procedure is repeated for all 5 random start locations to route them to the goal state. We obtain the top 20 routes for each location. Table 6.2 provides the average values for the 20 routes obtained. Furthermore, the results of average values of the evaluation parameters are listed in table 6.3. The column that gives the percentage of shared road segments shows the quality of roads obtained by the proposed algorithm. A higher percentage indicates a higher repetition of road segments. When the values of the routes obtained by PPA are compared with the values obtained by the standard Dijkstra's algorithm the travel time and distance have higher averages as expected. However, observing the mean elevation values obtained by PPA we can clearly observe that there are a lot of alternative routes that have higher elevation than the shortest path, thus ensuring safety.This is clearly illustrated in table 6.4, which lists the minimum elevations of the 20 routes between start ID 3 and Goal state 1.

Figures 6.4, 6.5, 6.6, 6.7, 6.8, and 6.9 give the results of visualization of the top 20 routes obtained at 5 different cost penalties using the path penalty algorithm for start ID 3 to goal. It can be observed from these figures clearly shows that increasing the cost penalty leads to creation of more disjoint routes. While it is clear that there were about 11 distinct paths between the two given points, it is important to observe the minimum elevations of these routes. We observe that the minimum elevation has

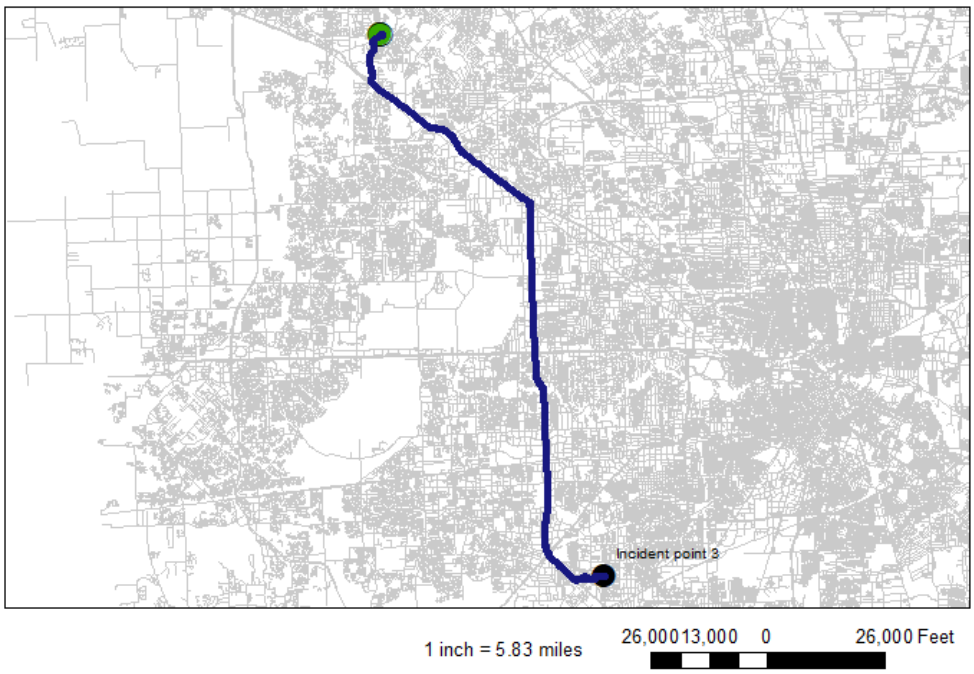


Figure 6.4: Routes obtained by PPA at cost = 0

the highest peaks at the two highest cost penalties as observed in figure 6.10 which graphically illustrates the values of minimum elevation listed in table 6.4 . These peaks are observed in routes 6, 11 and 16.

6.2.2.2 Randomized Segment Penalty Algorithm

The randomized segment penalty algorithm (or RSPA) is an variation of the path penalty algorithm. In this algorithm we obtain a least cost path and then randomly choose one segment to assign a high penalty to obtain subsequent routes. Figures 6.11, 6.12 and 6.13 give the results of visualization of the top 20 routes obtained at 3 different cost penalties using the path penalty algorithm between start ID 3 and goal. Figure 6.14 marks the different segments chosen for random penalizing when

Start ID	Goal ID	Name of Route	Total Time (min)	Total Length (miles)	Minimum Elevation(feet)	Mean Elevation(feet)
1	1	Incident point 1 - Goal	10.62	6.02	43.86	44.01
2	1	Incident point 2 - Goal	23.10	20.74	25.93	26.17
3	1	Incident point 3 - Goal	30.69	29.3	22.12	23.01
4	1	Incident point 4 - Goal	47.37	34.23	43.86	44.01
5	1	Incident point 5 - Goal	34.30	32.38	10.56	10.65

Table 6.5: Averages of RSPA results

Start ID	Goal ID	Speed Limit (miles/hr)	Percentage of shared road segments
1	1	32	96%
2	1	45.2	98%
3	1	43	98.3%
4	1	51	97.17%
5	1	47.3	98.9%

Table 6.6: Averages of RSPA Evaluation Measures

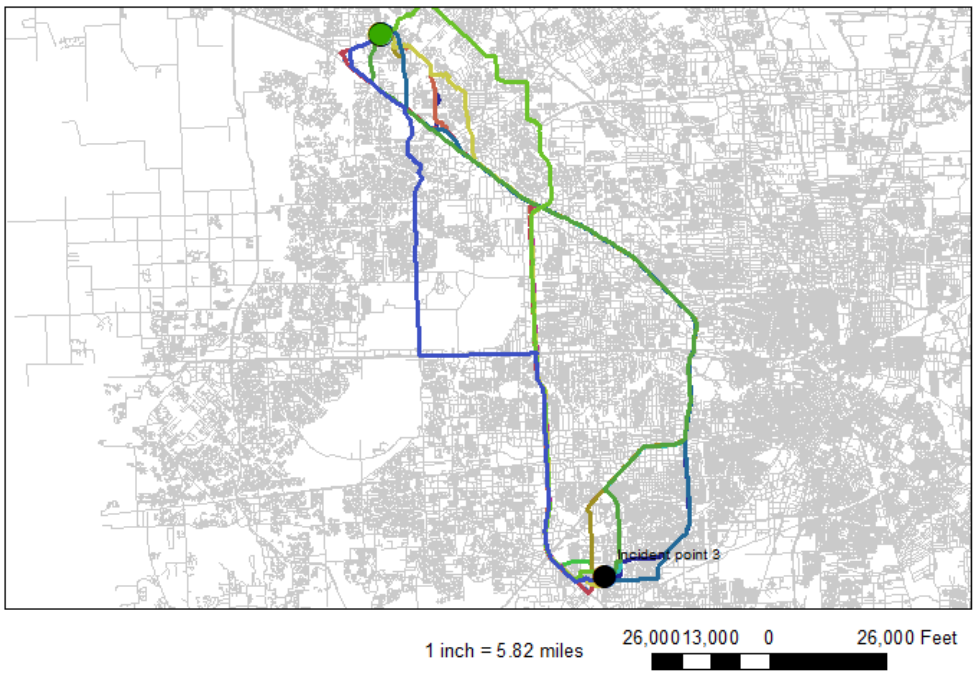


Figure 6.5: Routes obtained by PPA at cost = 1

the cost is set to 1.5. Table 6.5 lists the average values of different route evaluation parameters for the 20 routes obtained. Further, 6.6 lists the quality measures of the routes obtained.It is important to note that each of the routes obtained at the three proposed penalties give routes that are at least 95% identical. Thus, this algorithm is only useful to obtain routes that have slight alterations in selection of road segments to simulate localized road blockage issues.

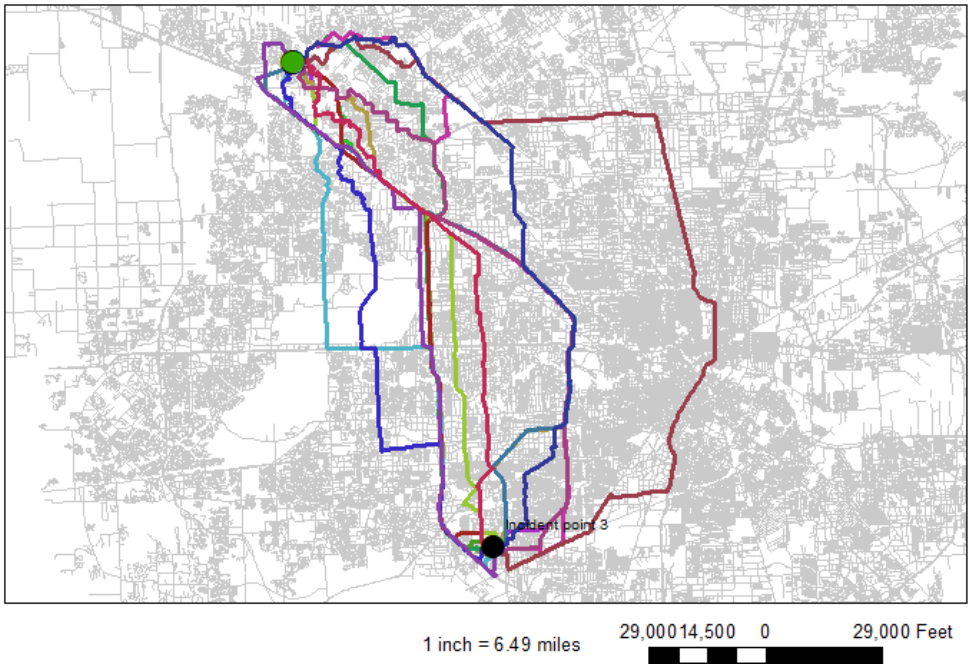


Figure 6.6: Routes obtained by PPA at cost = 3

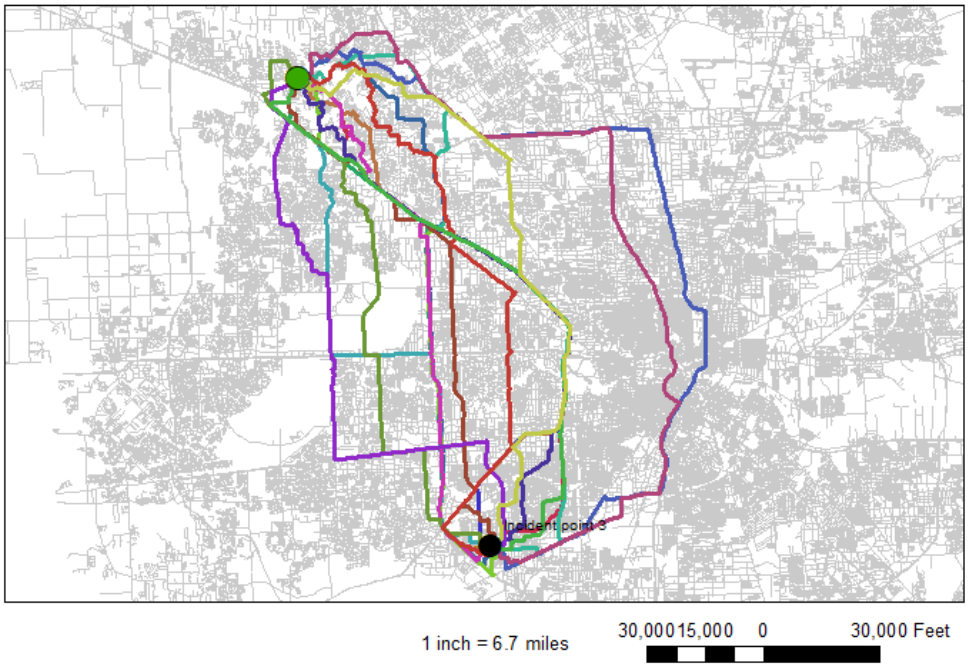


Figure 6.7: Routes obtained by PPA at cost = 5

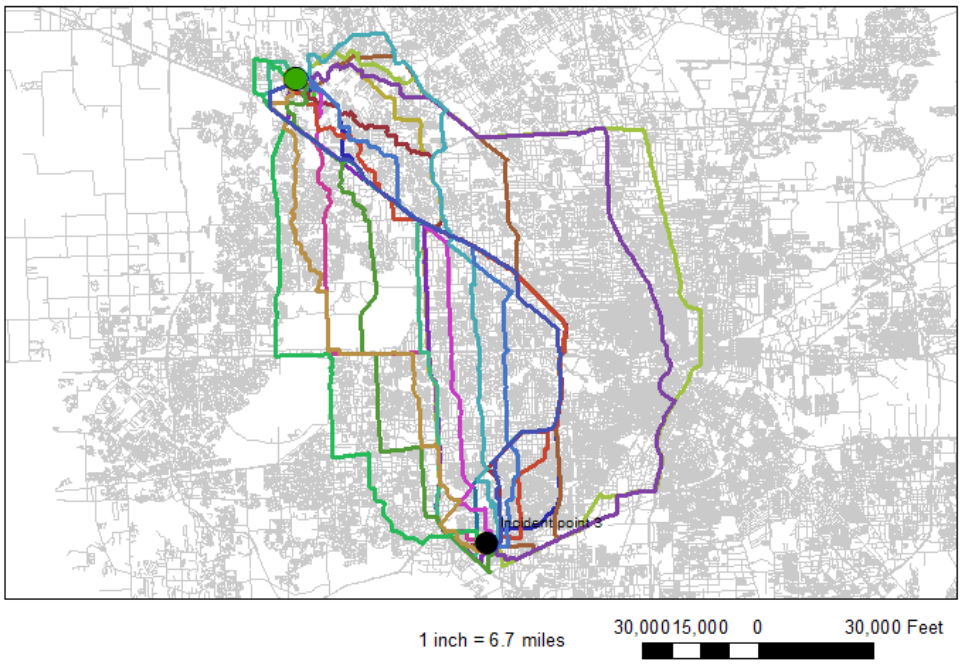


Figure 6.8: Routes obtained by PPA at cost = 7

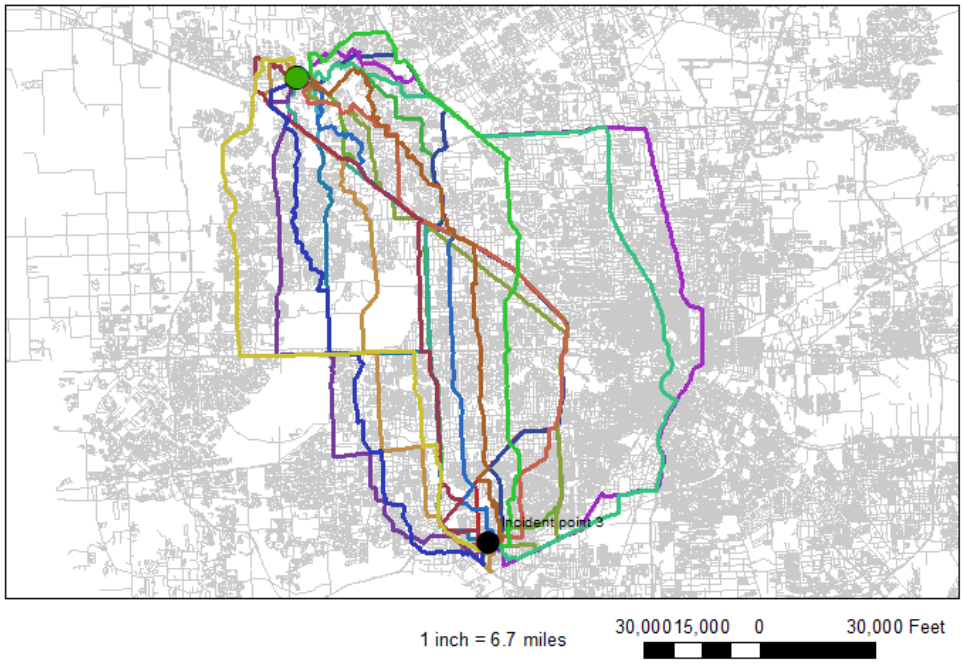


Figure 6.9: Routes obtained by PPA at cost = 9

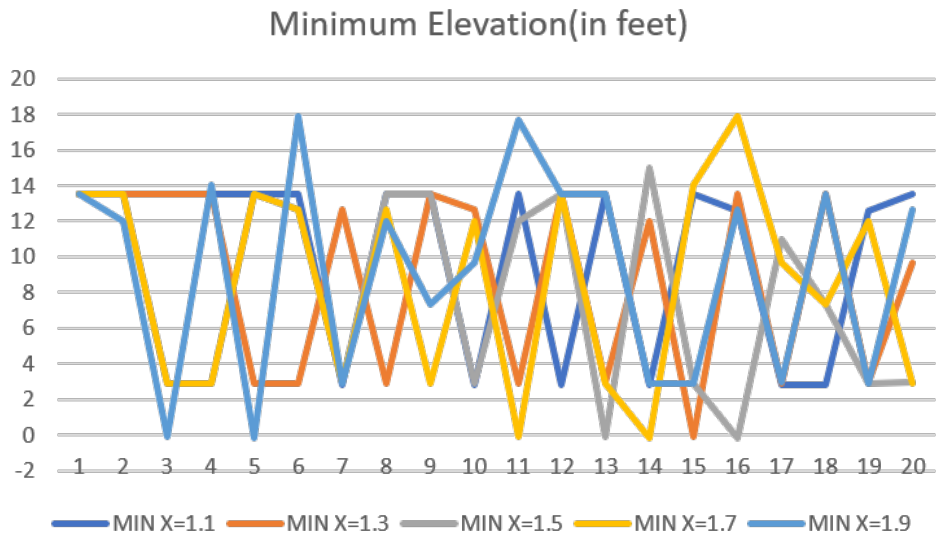


Figure 6.10: Minimum elevation measurements of the top 20 routes by PPA for start ID 3 and goal state

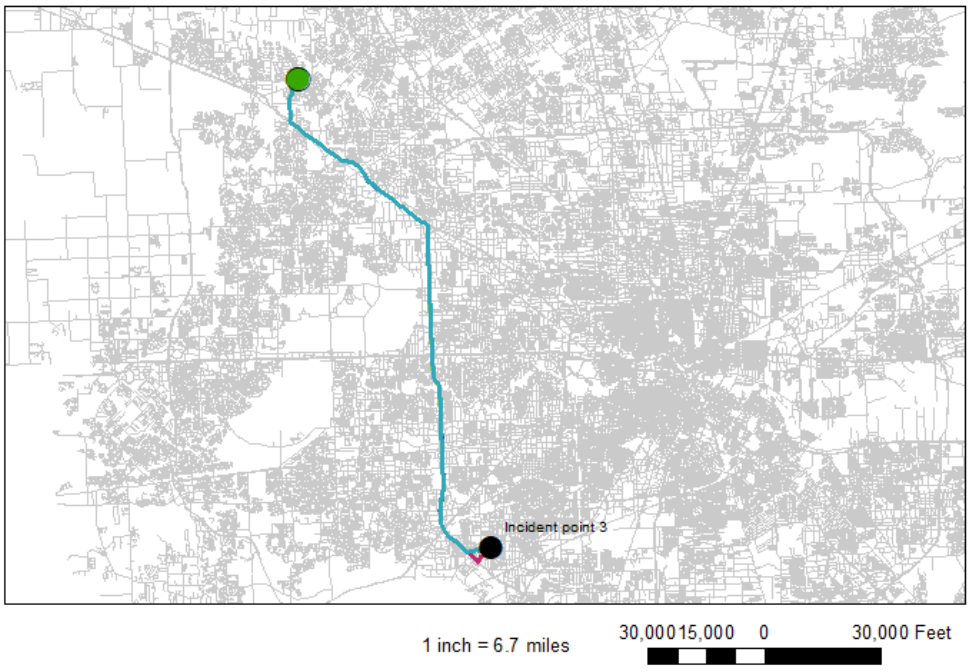


Figure 6.11: Routes obtained by RSPA at cost penalty= 1.5

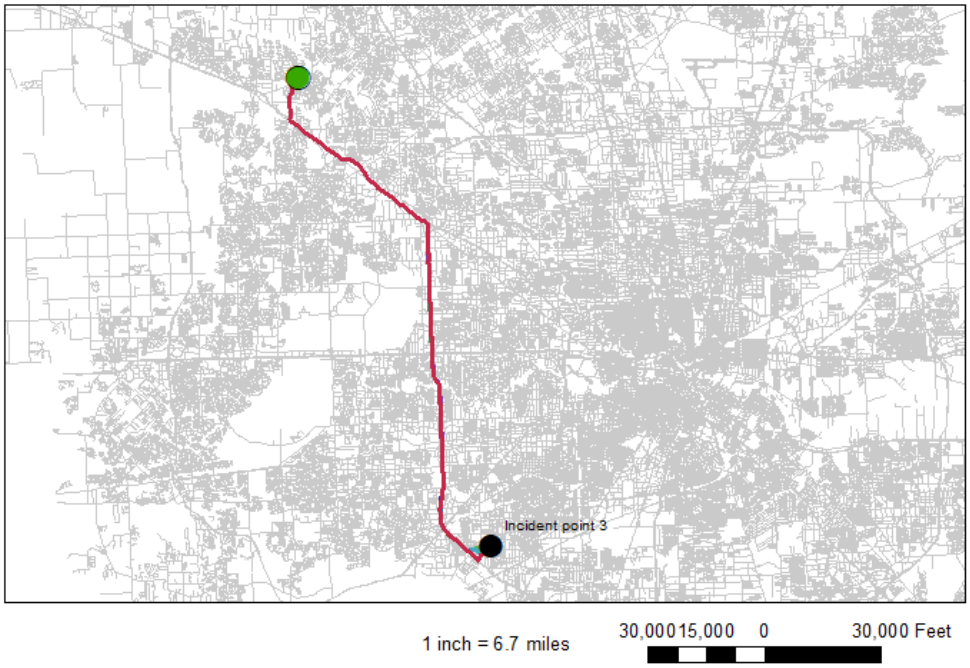


Figure 6.12: Routes obtained by RSPA at cost penalty = 5

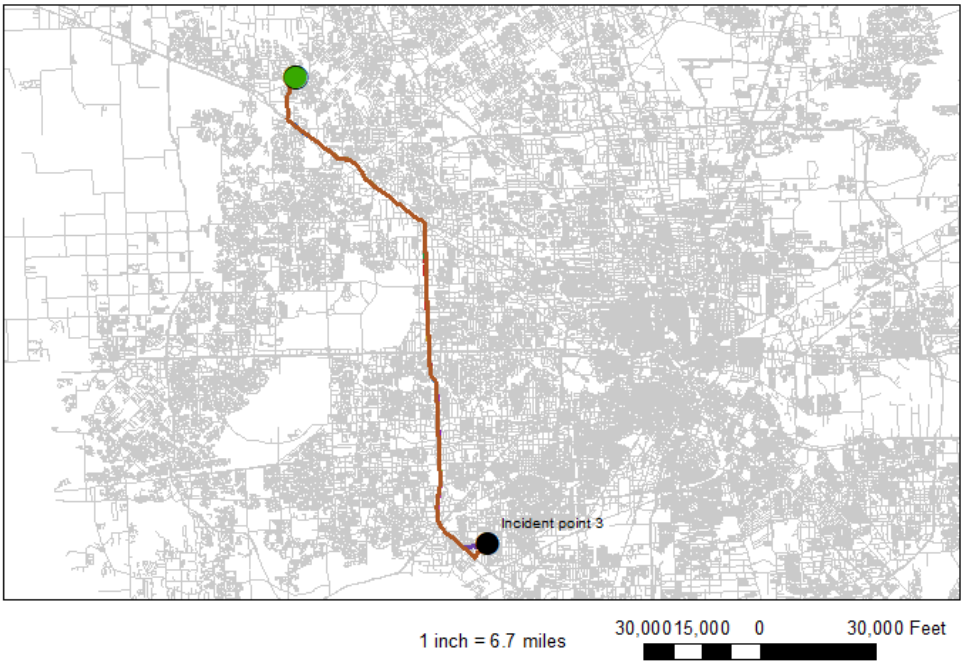


Figure 6.13: Routes obtained by RSPA at cost penalty = 10

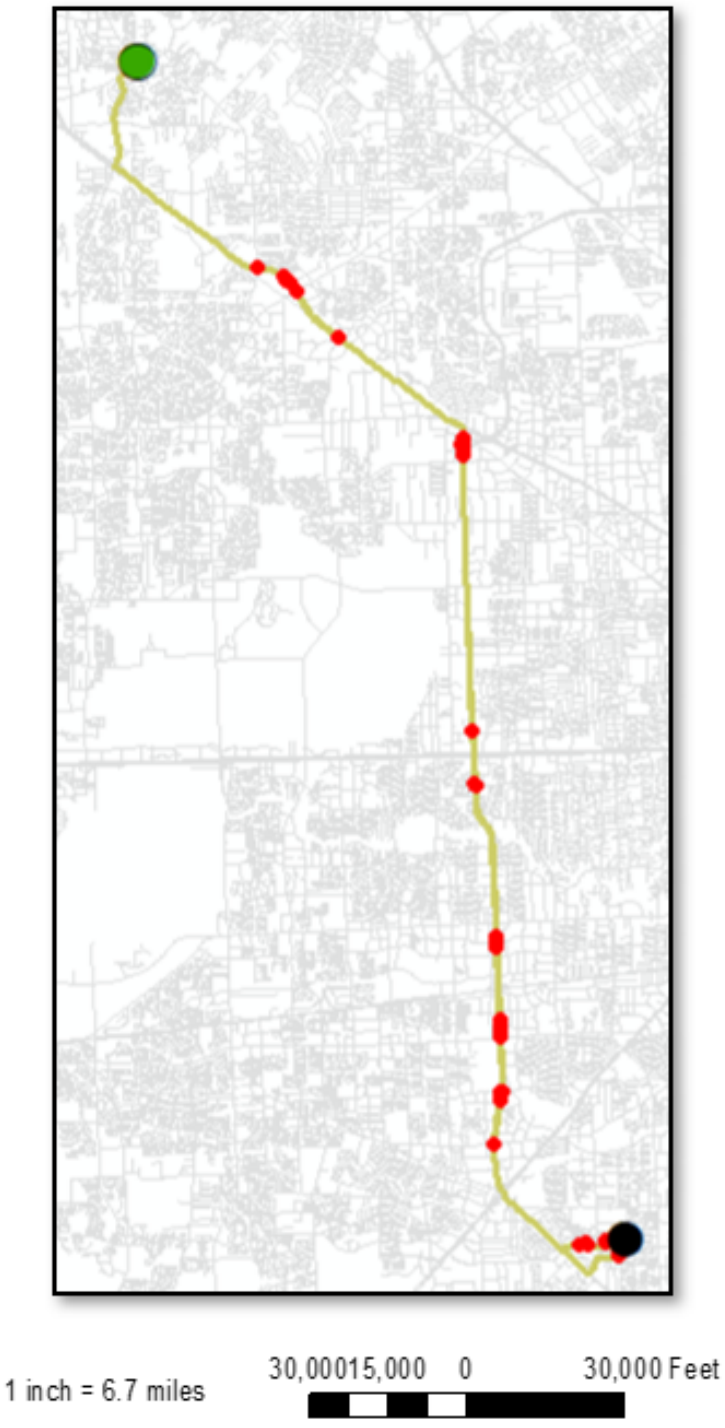


Figure 6.14: Segments chosen by Randomizer as cost barriers at X=1.5 for RSPA

Chapter 7

Conclusion

In this thesis we propose a unique approach to multiple route selection in the presence of flooding. This thesis initially began with the study of path planning leveraging the capabilities of GIS. We studied different planning frameworks and their applications. We tried to develop a street network navigation planning problem using PRODIGY. PRODIGY is an integrated planning and learning system that can learn control rules, conduct experiments to acquire new knowledge, generate abstraction hierarchies and use analogical reasoning to recognize and exploit similarities between problems [13]. However, testing PRODIGY revealed several drawbacks in its usage. First, PRODIGY was available exclusively in UNIX environments and ArcGIS was available only for Windows environments. Further, PRODIGY was not capable to handle the size and complexity of road dataset of a city such as Houston. Further investigation lead to discovery of significant memory usage for the planner to successfully store any new knowledge or hierarchies. For all these reasons, the approach

of using the concepts of AI Planning was abandoned and replaced by AI Heuristic Search.

Path Search algorithms most useful for transportation networks were studied. It was found that the most common methodology of route selection is by emulating graph traversal and that the most common algorithms for routing included Dijkstra's and A*. Most applications use these algorithms with slight modifications such as Physical-A*(PHA*) routing algorithm developed in [12]. We then studied the different evacuation and navigation techniques currently available for routing during floods. Regulations issued by the United States Government focused on moving to higher grounds and mass evacuations only when ordered. Further, the public was advised to do the navigation by keeping track of the current road and weather conditions using radio, government websites, etc. The research in this domain includes a detailed study of timeline modelling of flooding evacuation operations as stated in [6]. While this is integral in preparation of flood evacuation plans, it does not aid the actual navigation. Emane et al. in [15] present a Multi Agent System that provide safe routes during flooding. This is done by associating a road damage factor to each road segment. However, this approach does not provide any support during actual evacuations when it is not possible to assess road damage in real time. Furthermore, the approach proposed by Chakraborty et al. in [3] focuses on a randomized approach to alternative path finding using the concepts of Genetic Algorithms. The key idea is to find alternative paths by randomly selecting road segments from previously obtained routes. Thus, to the best of our ability we did not find any navigation algorithms that can support safer navigation in real time using minimal dynamically affected

characteristics.Further, we also did not find any concrete evaluation parameters that enable in quality assessment of the obtained routes.

To solve this issue, we proposed a framework that can find multiple routes and their associated features. This framework provides a set of top k routes given a start and goal state. The key idea is to find a cost optimal solution for two locations using Dijkstra's algorithm. We then penalize the found solution by increasing the traversal cost of one segment or the whole path, forcing the search algorithm to find alternative solutions. Once multiple routes are obtained, our approach proceeds to evaluation of quality and safety of the obtained routes. The two proposed variations of our approach have very specific purposes. The first approach, that penalizes one segment provides with k alternative routes that are very similar to each other. This is extremely useful in cases where there is intensive localized flooding. The second approach which penalizes the whole path provides top k disjoint solutions. This is useful during high flood water scenarios across the whole street network.

There are several additions that would make the proposed framework more robust. The most important of these is the inclusion of real time traffic data into the system. We plan to integrate this into the framework in the future. Further, historic and real time data from flood gauges across the city would be extremely helpful in modelling floods across individual street segments. We plan to use this data while selection of road segments to improve the safety of the routes. Finally, the current framework was developed and tested on a single instance of ArcGIS which lead to significant time constraints. It took as long as 15 minutes to compute 20 alternative

paths between a given start and goal location. We are investigating the possibility of developing this framework in a distributed environment that will significantly improve the route calculation times.

Bibliography

[1] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

[2] berkeley.edu. *GIS Data Formats*, (accessed July 6, 2018). `http://gif.berkeley.edu/documents/GIS_Data_Formats.pdf`.

[3] B. Chakraborty. Ga-based multiple route selection for car navigation. In *Asian Applied Computing Conference*, pages 76–83. Springer, 2004.

[4] I. A. Chandio, A. N. B. Matori, K. B. WanYusof, M. A. H. Talpur, S. H. Khahro, and M. R. M. Mokhtar. Computer application in routing of road using least-cost path analysis in hillside development. *Research Journal of Environmental and Earth Sciences*, 4(10):907–911, 2012.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[6] S. O. ESM, P. C. OAM, and B. Davies. Timeline modelling of flood evacuation operations. *Procedia Engineering*, 3:175–187, 2010.

[7] esri. *arcgis network analyst*, (accessed July 6, 2018). `https://www.esri.com/en-us/arcgis/products/arcgis-network-analyst/overview`.

[8] esri. *Network elements*, (accessed July 6, 2018). `3.http://desktop.arcgis.com/en/arcmap/latest/extensions/network-analyst/network-elements.htm`.

[9] esri. *what is dtop*, (accessed July 6, 2018). `http://edndoc.esri.com/arcobjects/9.2/cpp_vb6_vba_vcpp_doc/shared/desktop/get_started/what_is_dtop.htm`.

[10] esri. *what is gis*, (accessed July 6, 2018). `https://www.esri.com/en-us/what-is-gis/overview`.

[11] D. U. I. I. P. N. E. J. M. E. C. H. K. E. K. E. T. M. N. O. c. Esri, HERE and the GIS User Community. *World Street Map*.

[12] A. Felner, R. Stern, and S. Kraus. Pha*: Performing a* in unknown physical environments. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 240–247. ACM, 2002.

[13] E. Fink and M. M. Veloso. Prodigy planning algorithm. 1994.

[14] K. Z. Haigh, J. R. Shewchuk, and M. M. Veloso. Route planning and learning from execution. In *Preprints of the AAAI 1994 Fall Symposium on Planning and Learning: On to Real Applications, New Orleans, LA*, 1994.

[15] S. Imane, C. Loubna, E. Mostafa, and E. Mohammed. Intelligent evacuation system for flood disaster. In *Intelligent Distributed Computing VI*, pages 205–210. Springer, 2013.

[16] M. Loidl, G. Wallentin, R. Cyganski, A. Graser, J. Scholz, and E. Haslauer. Gis and transport modelingstrengthening the spatial perspective. *ISPRS International Journal of Geo-Information*, 5(6):84, 2016.

[17] A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. Basalamah. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044*, 2017.

[18] R. of the University of Minnesota. *Understanding the Power of GIS Mapping*, (accessed July 23, 2018). `http://blog-family-matters.extension.umn.edu/2016/07/understanding-power-of-gis-mapping.html`.

[19] P. Scerri, B. Kannan, P. Velagapudi, K. Macarthur, P. Stone, M. Taylor, J. Dolan, A. Farinelli, A. Chapman, B. Dias, et al. Flood disaster mitigation: A real-world challenge problem for multi-agent unmanned surface vehicles. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 252–269. Springer, 2011.

[20] tomtom. *Totom Traffic Index*, (accessed July 6, 2018). `https://www.tomtom.com/en_gb/trafficindex/city/houston`.

[21] M. Veloso, J. Carbonell, A. Perez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The prodigy architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[22] S. Wang, S. Djahel, J. McManis, C. McKenna, and L. Murphy. Comprehensive performance analysis and comparison of vehicles routing algorithms in smart cities. In *Global Information Infrastructure Symposium, 2013*, pages 1–8. IEEE, 2013.

[23] Wikipedia. *Graph theory*, (accessed July 6, 2018). `https://en.wikipedia.org/wiki/Graph_theory`.

[24] Wikipedia. *Shortest path problem*, (accessed July 6, 2018). `https://en.wikipedia.org/wiki/Shortest_path_problem`.